Optimisation non différentiable et optimisation stochastique : de la théorie aux applications

Welington de Oliveira

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

UNIVERSITÉ PARIS 1
PATHÉON SORBONNE

Mémoire de Synthèse
présenté par

Welington Luis de Oliveira

pour l'obtention d'une

Habilitation à diriger des recherches

Spécialité : Mathématiques Appliquées

Optimisation non différentiable et optimisation stochastique : de la théorie aux applications

soutenu le : 19/12/2018
devant un jury composé de :

M. Michel Théra	Président
M. René Henrion	Rapporteur
M. Georg Pflug	Rapporteur
M. Philippe Bich	Examinateur
M. Stephane Gaubert	Examinateur
Mme. Nadia Maïzi	Examinateur
Mme. Claudia Sagastizábal	Examinateur
M. Jean-Marc Bonnisseau	Garant

Ce travail est dédié à mon épouse Valentina.

Remerciements

Je souhaiterais exprimer toute ma gratitude au personnel du Centre de Mathématiques Appliquées (CMA) de MINES ParisTech pour l'accueil qu'il m'a réservé depuis mon arrivée et son soutien permanent qui a grandement contribué à mon intégration. En particulier, je voudrais remercier la directrice du CMA Nadia Maïzi pour son appui et ses encouragements à rédiger ce manuscrit, la responsable administrative et financière du CMA Catherine Auguet-Chadaj pour son aide inconditionnelle, et mes collègues Valérie Roy et Matthieu Denoux pour le concours qu'ils m'ont apporté dans la relecture de la partie française de ce mémoire.

Je suis également très reconnaissant à tous mes co-auteurs, en particulier Wim van Ackooij, Erlon Finardi, Antonio Frangioni, Claude Lemaréchal, Jérôme Malick, Claudia Sagastizábal et Mikhail Solodov que je tiens à remercier pour nos échanges techniques enrichissants et l'amitié qu'ils veulent bien m'accorder.

Je suis très sensible à l'honneur que me font professeurs René Henrion et Georg Pflug en acceptant d'être rapporteurs de mon travail, le professeur Michel Théra de présider le jury de mon HDR, et les professeurs Philippe Bich, Jean-Marc Bonnisseau, Stephane Gaubert, Nadia Maïzi et Claudia Sagastizábal en acceptant d'être examinateurs.

Enfin, j'adresse mes plus sincères remerciements à mes familles brésilienne et italienne pour leur soutien et leurs encouragements.

Welington Luis de Oliveira
École Nationale Supérieure des Mines de Paris - Mines ParisTech
Centre de Mathématiques Appliquées - CMA
1, rue Claude Daunesse, 06904, Sophia Antipolis, France
E-mail: welington.oliveira@mines-paristech.fr

Sophia Antipolis, décembre 2018

Abstract

This document is devoted to theoretical and practical aspects of two subfields of the mathematical programming, namely nonsmooth optimization and stochastic programming. In general, real-life stochastic programming models give rise to large-scale optimization problems that can only be solved with the help of decomposition techniques and specialized algorithms. Invariably, techniques such as Benders decomposition, Lagrangian relaxation, two-stage or multistage decomposition yield objective functions that are not only nonsmooth but also difficult to evaluate. For this reason, efficient nonsmooth optimization methods must come into play. This document highlights some of my contributions to these key subfields for decision-making in real-life optimization problems, such as those coming from the industry of energy.

Résumé étendu

Ce mémoire présente des aspects spécifiques de mes recherches et donne un aperçu de mes contributions sur des sujets pratiques et théoriques de la programmation (optimisation) mathématique. Motivé par des applications réelles de l'industrie de l'énergie, le cœur de mes recherches réside dans les aspects algorithmiques de l'optimisation non différentiable (non lisse) et de la programmation stochastique, mais a également une intersection avec des sujets plus théoriques tels que l'analyse de complexité, l'analyse variationnelle et la convergence forte dans les espaces de Hilbert de certaines méthodes de faisceaux.

Le document est rédigé en anglais, à l'exception de cette partie introductive qui présente l'organisation du mémoire et résume le chapitre 1 en mettant en évidence mes contributions à l'optimisation non lisse, à la programmation stochastique et à leurs applications aux problèmes de la gestion de l'énergie.

La première section ci-dessous présente une vue générale des domaines de mes recherches, et la deuxième partie donne un aperçu des contributions scientifiques rapportées dans les chapitres 2, 3, 4, 5 et 6. Cette partie introductive adopte un style concis et conversationnel, évitant une quantité significative de détails techniques. L'objectif est de rendre le contenu accessible non seulement aux experts sur des sujets de l'optimisation non lisse et/ou stochastique, mais aussi à un public plus large. Les autres chapitres présentent cinq de mes articles sur l'optimisation non lisse et la programmation stochastique qui ont été publiés dans des revues spécialisées de programmation mathématique au cours des années 2016, 2017 et 2018. Ces contributions, présentés en détail dans les chapitres suivants, ont été choisies pour donner une idée de mes divers sujets de recherche.

Les chapitres 2 et 3 traitent de l'optimisation déterministe non lisse : le premier discute de l'optimisation convexe en présentant de nouvelles extensions des méthodes dites de niveau [50], et le second considère l'optimisation non convexe et étudie une classe d'algorithmes avec une force d'inertie pour résoudre des programmes DC (Difference de fonctions Convexes) [51].

Le chapitre 4 décrit le problème de planification à long terme de la conception et de l'gestion du réseau brésilien de gaz naturel étudié dans [32]. Une telle application aussi importante est modélisée comme un programme linéaire stochastique à deux étapes et résolue en combinant une technique de décomposition, un algorithme de faisceaux, et des techniques de réduction de scénarios.

Les chapitres 5 et 6 traitent de l'optimisation sous contrainte en probabilité : le premier présente des résultats sur la convexité éventuelle de problèmes sous contrainte en probabilité structurée par Copulæ [11], et le second rapporte le travail [4] sur des techniques d'optimisation basées sur le concept de *p-efficient point* et des méthodes de faisceaux inexactes.

Enfin, le chapitre 7 conclut avec quelques remarques et des directions futures de mes recherches. Une liste de toutes mes activités de recherche depuis mon doctorat, sous la forme d'un Curriculum Vitæ étendu, est donnée en annexe.

Programmation mathématique : champs d'action

La programmation mathématique est la branche des mathématiques concernée par la théorie et les méthodes permettant de trouver des *extrema* de fonctions sur des ensembles d'espaces vectoriels. Le terme "programmation mathématique" est lié au fait que l'objectif de résoudre un problème mathématique (de trouver un maximum ou un minimum d'une fonction sur un ensemble admissible) est de choisir un programme/plan d'action. Par exemple, une entreprise dans le domaine de l'énergie cherche un programme d'action pour générer de l'électricité au coût minimum pour répondre à une demande de charge.

La formulation des problèmes de programmation mathématique (ou simplement problèmes d'optimisation) considérée dans ce document est la suivante :

$$\min_{x\in X} f(x) \quad \text{s.t.} \quad \mathfrak{c}(x) \leq 0\,, \tag{0.0.1}$$

où $X \neq \emptyset$ est contenu dans un ensemble ouvert O d'un espace de Hilbert $\mathcal{H}$ (dans la plupart de ce

document $\mathcal{H}$ est simplement l'espace Euclidien $\mathbb{R}^n$) et $f, \mathfrak{c} : O \to \mathbb{R}$ sont fonctions convexes données. Dans le contexte de l'optimisation non lisse, il n'y a pas de perte de généralité en considérant une fonction de contrainte scalaire $\mathfrak{c}$, car elle peut être définie comme le maximum de toutes les fonctions de contrainte.

On distingue habituellement les branches suivantes de la programmation mathématique : linéaire, quadratique, convexe, non lisse, stochastique, entière (mixte)/discrète, DC, etc. Naturellement, un problème d'optimisation donné peut correspondre à plusieurs branches : par exemple, un problème de programmation stochastique peut être non lisse et convexe en même temps. En fait, c'est dans cette intersection des branches de la programmation mathématique que se trouve une partie importante de ma recherche [4, 23, 148, 199]. La plupart de mes travaux traitent des problèmes d'optimisation convexe non lisse, à savoir la formulation (0.0.1) où X est un ensemble convexe et où les fonctions f et $\mathfrak{c}$ sont convexes, e.g. [5, 9, 21, 45, 48, 50, 148, 149]. D'autres de mes publications considèrent les problèmes convexes de programmation en nombres entiers mixtes (MINLP) : problème (0.0.1) avec des fonctions convexes mais où X est un ensemble à nombres entiers mixtes [53, 150, 203]. Récemment, mes recherches ont été étendues à un large éventail de problèmes dans lesquels X est convexe, mais f et $\mathfrak{c}$ sont des fonctions DC [51, 201].

Pourquoi étudier l'optimisation non lisse ? La programmation non lisse étudie la résolution des problèmes d'optimisation sous la forme (0.0.1) où la fonction objectif f ou la contrainte $\mathfrak{c}$ n'est pas continûment différentiable. Pour résoudre de tels problèmes, il est nécessaire de considérer un objet mathématique spécial : le sous-différentiel d'une fonction convexe f, noté $\partial f(x)$, qui est un concept fondamental dans l'analyse convexe. Si f est lisse (c'est-à-dire continûment différentiable) en un point donné x, alors le sous-différentiel de f en x est un singleton et coïncide avec le gradient $\nabla f(x)$. Sinon, dans le contexte convexe, le sous-différentiel de f en x est un ensemble donné par

$$\partial f(x) := \{g \in \mathcal{H} : \ f(y) \geq f(x) + \langle g, y - x\rangle \ \ \forall\, y \in \mathcal{H}\}\,.$$

Cette définition dit que $\partial f(x)$ est composé des pentes des hyperplans supportant l'épigraphe de f en $(x, f(x))$. Chaque élément g de $\partial f(x)$ est appelé un *sous-gradient*.

La principale raison d'examiner les méthodes d'optimisation non lisse est la suivante : la non-différentiabilité des fonctions exclut l'application des méthodes d'optimisation lisse. Souvent en pratique, les (sous) gradients des fonctions impliquées ne sont pas calculés exactement. En optimisation différentiable (lisse), les gradients sont fréquemment obtenus à partir des valeurs fonctionnelles par des différences finies. Cette approche n'est valide que dans le cas lisse, car $f'(x; d) = \langle \nabla f(x), d\rangle$ est linéaire en d et peut être approché par des quotients des différences finies. Lorsque f n'est pas différentiable, la fonction multivaluée $x \to \partial f(x)$ n'est pas continue. En conséquence, les quotients des différences finies n'appartiennent pas nécessairement au sous-différentiel, même pas à la limite [29, page 125]. En fait, la non-différentiabilité empêche même l'application des méthodes sans dérivées (algorithmes qui ne nécessitent pas de calcul de dérivées) : puisque l'analyse de la convergence de ces méthodes dépend fortement des hypothèses de différentiabilité [42].

De plus, de nombreuses applications motivent l'étude de la programmation non lisse. En effet, les problèmes de minimisation d'origine économique conduisent fréquemment à la manipulation de fonctions qui ne sont pas continûment différentiables. Très souvent, la non-différentiabilité est introduite artificiellement par les techniques de décomposition utilisées pour la résolution des problèmes "réels", qui sont à la fois de grande taille et de nature hétérogène. C'est le cas du problème de planification du réseau de gaz naturel présenté au chapitre 4.

Pourquoi étudier l'optimisation stochastique ? La programmation stochastique est la branche de la programmation mathématique dans laquelle on étudie la théorie et les méthodes pour résoudre des problèmes d'optimisation qui dépendent de paramètres incertains ayant des *distributions de probabilités connues*. L'optimisation stochastique est parfois appelée *optimisation sous incertitude*, mais cette dernière désignation devrait être évitée car elle est si large qu'elle englobe des problèmes d'optimisation

pour lesquels aucun modèle stochastique (avec une distribution de probabilité connue) n'est disponible. En raison de la présence de paramètres aléatoires, la théorie de la programmation stochastique combine les concepts d'optimisation avec la théorie des probabilités et des statistiques [181].

Les nombreuses applications motivent l'étude de la programmation stochastique : des modèles d'optimisation stochastique sont présents dans presque tous les domaines de la science et de l'ingénierie, de la gestion du système d'électricité, transport et à la finance, etc. De tels modèles sont généralement utilisés comme substituts aux formulations déterministes lorsque ces dernières se présentent comme insatisfaisantes ou inappropriées pour faire face à la prise de décision dans des situations réelles. En général, les modèles d'optimisation stochastique donnent lieu à des problèmes d'optimisation de grande taille qui ne peuvent être résolus qu'à l'aide de techniques de décomposition et d'algorithmes spécialisés. Souvent, des techniques telles que la décomposition de Benders, la relaxation Lagrangienne, la décomposition à deux où à plusieurs étapes rendent des fonctions objectives qui ne sont pas seulement *non lisse* mais aussi *difficile à évaluer*. Ces deux difficultés sont présentes dans les deux sous-domaines de la programmation stochastique, à savoir la *programmation stochastique avec recours* et la *programmation sous contrainte en probabilité*.

Dans les problèmes d'optimisation avec recours, il faut résoudre plusieurs sous-problèmes pour estimer la fonction non lisse f (généralement donnée sous une forme d'espérance mathématique), et dans les problèmes sous contrainte en probabilité, la fonction $\mathfrak{c}$ dépend d'une mesure de probabilité qui est calculée via l'intégration numérique multidimensionnelle et/ou la simulation. Dans les deux cas, évaluer exactement les fonctions dans des temps de calcul raisonnables est une tâche difficile en général. Cette difficulté est encore plus critique lorsque l'on considère des problèmes d'optimisation de grande taille, comme ceux provenant de l'industrie de l'énergie.

Problème de la gestion de l'énergie : comme application et source d'inspiration. Le problème de la gestion de l'énergie consiste à utiliser efficacement des ressources pour répondre aux besoins énergétiques. Il rassemble la planification et la gestion d'unités de production et de consommation d'énergie. À titre d'exemple, dans les systèmes électriques, il faut générer de l'électricité pour répondre à la demande en temps réel, ce qui implique une planification appropriée et éventuellement une augmentation des moyens de production. Bien que la minimisation des coûts et la maximisation des revenus soient les objectifs les plus courants, la conservation des ressources et la protection environnementale reviennent de plus en plus fréquemment dans les débats tandis que de nouvelles lois relatives à l'évolution des systèmes énergétiques entrent en jeu.

Dans la catégorie des problèmes de la gestion de l'énergie, l'importance des méthodes de programmation mathématique a été reconnue depuis longtemps. Avec la croissance des sources d'énergie renouvelable intermittentes, le besoin de gérer l'incertitude dans de tels problèmes d'optimisation est devenu primordial pour la quête de rentabilité et d'une plus grande fiabilité du système. Dans ce nouveau paradigme, comment pouvons-nous gérer le caractère aléatoire des sources renouvelables de production d'électricité ? La programmation stochastique offre des réponses à cette question, pour cette raison, les méthodes d'optimisation stochastique sont devenues des outils essentiels dans l'industrie de l'énergie.

Les problèmes d'énergie réels engendrent des problèmes d'optimisation de grande taille qui, en général, ne peuvent être résolus qu'à l'aide de techniques de décomposition. Comme mentionné ci-dessus, de telles techniques conduisent à des fonctions objectif qui sont non seulement non lisses mais également difficiles à évaluer. En plus, très souvent, l'hypothèse de convexité est aussi absente. Ces caractéristiques soulèvent des problèmes d'optimisation présentant un niveau de difficulté marquant, parfois prohibitif dans le cadre des problèmes de grande taille. En conséquence, de nouvelles techniques d'analyse variationnelle sont nécessaires pour améliorer les méthodes de résolution dans ce contexte.

Un aperçu des travaux rapportés dans ce mémoire

Cette section présente un aperçu de cinq des mes travaux de recherches sur l'optimisation non lisse et la programmation stochastique, et met en évidence mes contributions dans ces domaines. La section commence par une discussion sur les algorithmes pour l'optimisation non lisse en présentant les méthodes de faisceaux, la programmation DC, et finit par les programmes stochastiques sous contrainte en probabilité.

Optimisation non lisse. Comme nous l'avons déjà mentionné, la non différentiabilité des fonctions impliquées empêche l'application des méthodes d'optimisation différentiable, y compris celles qui ne nécessitent pas de calcul des dérivées. Les problèmes d'optimisation non lisse nécessitent des méthodes spéciales tenant compte de la discontinuité du sous-différentiel. Parmi ces techniques spéciales, les méthodes du sous-gradient et des plans sécants sont peut-être les approches les plus connues pour les problèmes d'optimisation non lisse. De telles méthodes sont également connues pour avoir une convergence lente (dans certains cas, le nombre de points d'essai générés par la méthode des plans sécants augmente de façon exponentielle avec la dimension du problème [142, pp. 158-160]). Les méthodes de faisceaux [29] sont des algorithmes particulièrement appropriés pour l'optimisation convexe non lisse lorsque la précision de la solution et la fiabilité sont une priorité.

Les méthodes de faisceaux pour l'optimisation convexe.
Pour le moment, concentrons-nous sur le problème (0.0.1) sans la contrainte non linéaire $\mathfrak{c}$ et supposons que f est une fonction convexe. De plus, nous supposons qu'un oracle (exact) pour f est disponible : pour tout point donné $x \in X$, l'oracle nous fournit $f(x)$ et un sous-gradient $g \in \partial f(x)$.

En ayant collecté un *faisceau* d'information $\{(f(x_1), g_1), \ldots, (f(x_k), g_k)\}$ sur les points d'essai x_j, $j \in J_k := \{1, \ldots, k\}$, la convexité de f assure que le modèle des plan sécants

$$\check{f}_k(x) := \max_{j \in J_k} \{f(x_j) + \langle g_j, x - x_j \rangle\} \quad \text{satisfait} \quad \check{f}_k(x) \leq f(x) \quad \text{pour tout } x \in \mathcal{H}.$$

Avec un tel modèle à la main, la méthode des plans sécants de Kelley [103] définit un nouveau point d'essai comme une solution du programme maître

$$x_{k+1} \in \arg\min_{x \in X} \ \check{f}_k(x)\,, \tag{0.0.2}$$

qui est un problème de programmation linéaire si X est polyédrique. Un inconvénient majeur de la méthode devient évident : puisque le faisceau d'information indexé par les éléments de J_k croît avec le processus itératif, le sous-problème (0.0.2) devient de plus en plus difficile à résoudre. Contrairement à la méthode des plans sécants de Kelley, la plupart des méthodes de faisceaux ont une mémoire bornée : le faisceau d'information provenant d'oracle peut être borné, ce qui permet d'économiser de la mémoire informatique sans nuire à la convergence. Ceci est particulièrement intéressant pour les problèmes d'optimisation de grande taille. Sans faire de distinction de notation entre le modèle des plans sécants complet où $J_k = \{1, \ldots, k\}$ et un modèle éventuellement plus économique donné par un certain sous-ensemble $J_k \subset \{1, \ldots, k\}$, nous rappelons maintenant les principaux éléments des méthodes de faisceaux.

Les méthodes des faisceaux standard utilisent trois composants principaux : un modèle convexe $\check{f}_k$ qui (idéalement) est une sous-approximation de la fonction f; un centre de stabilité $\hat{x}_k$ qui est en général le "meilleur" point généré par le processus itératif; et un paramètre ($t_k > 0$, $f_k^{\text{lev}} \in \mathbb{R}$ ou $\rho_k > 0$) à mettre à jour à chaque itération k. Un nouveau point d'essai x_{k+1} d'une méthode de faisceaux dépend de ces composants, dont l'organisation définit plusieurs variantes :

la variante proximal

$$x_{k+1} := \arg\min_{x \in X} \ \check{f}_k(x) + \frac{1}{2t_k} \|x - \hat{x}_k\|^2 \tag{0.0.3}$$

la variante de niveau

$$x_{k+1} := \arg\min_{x \in X} \ \frac{1}{2} \|x - \hat{x}_k\|^2 \quad \text{s.t.} \quad \check{f}_k(x) \leq f_k^{\text{lev}} \tag{0.0.4}$$

la variante de région de confiance

$$x_{k+1} := \arg\min_{x \in X} \ \check{f}_k(x) \quad \text{s.t.} \quad \|x - \hat{x}_k\|^2 \leq \rho_k. \tag{0.0.5}$$

Il est bien connu que pour des paramètres (proximal t_k, niveau f_k^{lev} et région de confiance ρ_k) bien choisis et le même centre de stabilité $\hat{x}_k$, il est toujours possible de générer le même point d'essai en résolvant soit (0.0.3), (0.0.4) ou (0.0.5). Dans ce sens théorique, les trois approches peuvent être considérées comme équivalentes. Les détails de la mise en œuvre et des performances pratiques peuvent cependant être très différents, en particulier, parce que les paramètres sont mis à jour par des stratégies spécifiques à chacune des méthodes et que les règles correspondantes ne sont pas liées de manière directe.

D'autres variantes récentés de la méthode de faisceaux sont données dans [153], [48] et [50]. En particulier, le dernier article est rapporté dans le chapitre 2. Comme nous le verrons, [50] fournit l'analyse de convergence et des extensions de la méthode du rayon cible introduite brièvement à la fin de [153]. L'une des extensions proposées dans [50] est dédiée à la résolution de problèmes sous la forme de (0.0.1) avec la contrainte $\mathfrak{c}$ définie par une fonction continûment différentiable et fortement convexe $\mathfrak{s} : X \to \mathbb{R}$:

$$\mathfrak{c}(x) \leq 0 \quad \equiv \quad \mathfrak{s}(x) \leq \delta \, .$$

Chaque itération de la méthode du rayon cible proposée est donnée par :

$$x_{k+1} := \arg\min_{x \in X} \ \mathfrak{s}(x) \quad \text{s.t.} \quad f(x_j) + \langle g_j, x - x_j \rangle \leq f_{k,j}^{\text{lev}} \quad \forall \, j \in J_k \, . \tag{0.0.6}$$

Si le point d'essai donné ne satisfait pas la contrainte non linéaire $\mathfrak{s}(x) \leq \delta$, alors x_{k+1} est ignoré, $\min_{j \in J_k} \{f_{k,j}^{\text{lev}}\}$ devient une borne inférieure valide pour la valeur optimale de (0.0.1), et des nouveaux paramètres de niveau $f_{k+1,j}^{\text{lev}} > f_{k,j}^{\text{lev}}$ sont actualisés. Ceci est une règle originale et pratique pour définir des limites inférieures pour le problème (0.0.1). La formulation ci-dessus montre clairement le lien entre la méthode du rayon cible et les algorithmes de niveau : il suffit de prendre $f_{k,j}^{\text{lev}} := f_k^{\text{lev}}$ pour tous les $j \in J_k$, $\mathfrak{s}(x) := \frac{1}{2} \|x - \hat{x}\|^2$ et comparer avec (0.0.4). L'intérêt d'avoir des paramètres de niveau individuel (un paramètre par linéarisation) au lieu d'un paramètre de niveau unique pour le modèle est justifié par les "pentes" g_j, $j \in J_k$: si la pente générée par g_j est plus raide que celle émise par g_i, alors il pourrait être intéressant de prendre $f_{k,j}^{\text{lev}} < f_{k,i}^{\text{lev}}$ pour essayer de pousser x_{k+1} vers l'ensemble de solutions ou, dans l'autre cas, le pousser hors de l'ensemble admissible (e.g. $\mathfrak{s}(x_{k+1}) > \delta$), obtenant sans effort une nouvelle et meilleure estimation inférieure du problème (0.0.1). La méthode du rayon cible peut également être considérée comme une extension de la méthode du plan sécant avec le centre de Chebyshev [64].

L'article [50] prouve que l'algorithme donné possède une complexité d'itération indépendante de la dimension du problème et que le taux de convergence est optimal pour la minimisation d'une fonction convexe non lisse sur une boule Euclidienne, un simplex et d'autres domaines. Un mécanisme innovant permettant de borner la mémoire de la méthode est également proposé. En outre, [50] présente la première méthode de niveau pour la classe particulière d'optimisation à deux niveaux.

$$\min_{x \in \mathbb{R}^n} \ \mathfrak{s}(x) \quad \text{s.t.} \quad x \in \arg\min_{y \in X} f(y) \, .$$

La méthode du rayon cible et ses variantes sont examinées en détail dans le chapitre 2.

Programmation DC (Différence de fonctions Convexes).
La programmation DC constitue un sous-domaine important de la programmation non convexe qui reçoit beaucoup d'attention de la communauté de la programmation mathématique. Les problèmes de cette classe correspondent à la formulation (0.0.1) avec X un ensemble convexe, $f = f_1 - f_2$ et $\mathfrak{c} = c_1 - c_2$, où f_1, f_2, c_1 et c_2 sont des fonctions convexes. Certaines applications de la programmation DC incluent les problèmes sous contrainte en probabilité, les problèmes de gestion de l'énergie, l'analyse de regroupement et d'autres.

Les principaux avantages de la programmation DC sont qu'il s'agit d'une extension de la programmation convexe qui est suffisamment vaste pour couvrir presque tous les problèmes d'optimisation non convexe

(par exemple, toutes les fonctions *Lower-C^2* sont DC), tout en permettant l'utilisation d'outils puissants d'analyse convexe et d'optimisation convexe.

Comme pour l'optimisation non convexe et non lisse, de nombreuses définitions de points stationnaires existent pour l'optimisation DC non lisse. Quelle est la définition la plus forte ? Pouvons-nous calculer un point stationnaire ? Pour répondre à la première question, nous nous basons sur [155], qui prouve que la définition la plus forte est la B(ouligand)-stationnarité. En ignorant la contrainte non linéaire $\mathfrak{c}$ dans (0.0.1) et en se concentrant sur la formulation par contrainte convexe pour des raisons de simplicité, la définition de B-stationnarité est mieux connue sous le nom de d(irectionnel)-stationnarité :

$$\partial f_2(\bar{x}) \subset \partial f_1(\bar{x}) + N_X(\bar{x})\,,$$

où $N_X(\bar{x})$ est le cône normal de l'ensemble convexe X au point $\bar{x}$. Vérifier la B-stationnarité n'est pas une tâche facile lorsque f_2 est une fonction (convexe) non lisse générale. On se contentera donc d'une définition de stationnarité plus faible, appelée *criticité* :

$$\emptyset \neq \partial f_2(\bar{x}) \cap \partial f_1(\bar{x}) + N_X(\bar{x})\,.$$

(Notons que la criticité et la B-stationnarité coïncident lorsque f_2 est lisse).

Dans l'article [51], nous proposons une nouvelle classe d'algorithmes DC non monotones. Pour ce que nous savons, tous les algorithmes DC dans la littérature sont monotones et, en conséquence, sont facilement piégés par des points critiques de mauvaise qualité. Dans le but de calculer les points critiques qui sont également B-stationnaires (sans aucune hypothèse supplémentaire sur f_1 ou f_2), nous avons proposé un schéma algorithmique doté d'une procédure de force d'inertie semblable à la méthode *Heavy-Ball* de Polyak. Contrairement à l'algorithme DC classique de [193], nos approches ne nécessitent pas de résoudre les sous-problèmes convexes jusqu'à l'optimalité pour définir les points d'essai : seules les solutions inexactes du sous-problème convexe suivant suffisent

$$x_{k+1} \in \arg\min_{x\in X}\ f_1(x) - \langle g_2^k + \beta(x_k - x_{k-1}), x\rangle, \quad \text{avec} \quad g_2^k \in \partial f_2(x_k) \ \text{ et } \ \beta \geq 0 \ \text{ donnés.}$$

L'approche a été évaluée numériquement sur des modèles non convexes (de grande taille) de débruitage d'image. De tels modèles sont devenus des outils importants dans les systèmes de vision computationnelle. Plus de détails sur cette classe de méthodes sont présentés dans le chapitre 3.

Programmation stochastique. Presque tous les domaines de la science et de l'ingénierie utilisent des modèles d'optimisation stochastique. De tels modèles sont souvent des substituts aux formulations déterministes lorsque celles-ci se révèlent insatisfaisantes ou inappropriées pour prendre des décisions dans des situations réelles.

Programmation stochastique avec recours.
En programmation stochastique avec recours, la fonction objectif dans (0.0.1) est en général donnée par

$$f(x) = \varphi(x) + \mathcal{R}[Q(x,\omega)],$$

où ω est un vecteur aléatoire suivant une distribution de probabilité connue, $\varphi : \mathbb{R}^n \to \mathbb{R}$ est une fonction déterministe, $f : \mathbb{R}^n \times \Omega \to \mathbb{R}$ est une fonction dépendant à la fois de la variable de décision x et du vecteur aléatoire ω, et $\mathcal{R}$ est une mesure de risque [161], [181, chapitre 6]. Ci-dessous nous allons considérer le cas neutre au risque, c'est-à-dire, $\mathcal{R}$ est l'opérateur de l'espérance mathématique par rapport à la mesure de probabilité de ω : $\mathcal{R}[Q(x,\omega)] = \mathbb{E}[Q(x,\omega)]$.

Que signifie "recours" dans les programmes stochastiques ?
En règle générale, si l'on prend une décision *here-and-now* x qui se présente incorrecte lorsque le futur événement ω est révélé, il existe la possibilité de faire une correction (à un coût élevé) pour compenser une décision aussi défectueuse.

Dans la configuration de la programmation stochastique (linéaire) à deux étapes, la fonction de recours $Q(x,\omega)$ est la valeur optimale du problème (dual) de programmation linéaire

$$Q(x,\omega) := \begin{cases} \max & \langle b(\omega) - T(\omega)x, u\rangle \\ \text{s.t.} & W^\top u \leq q(\omega)\,, \end{cases} \tag{0.0.7}$$

pour des vecteurs $q(\omega)$, $b(\omega)$ et des matrices $T(\omega)$ et W donnés. Soit $u_{x,\omega}$ une solution du sous-problème ci-dessus : il est bien connu que, pour tout $x \in \mathcal{D}\mathrm{om}(f)$, le vecteur $g := \nabla\varphi(x) - \mathbb{E}[T(\omega)^\top u_{x,\omega}]$ est un sous-gradient de f en x, i.e., $g \in \partial f(x)$.

Les modèles d'optimisation stochastique emploient généralement une formulation par scénarios : le vecteur aléatoire ω est approximé par un échantillon de scénarios $\Omega := \{\omega^1, \omega^2, \ldots, \omega^N\}$ avec les probabilités associées $\pi_\omega > 0$, $\omega \in \Omega$, et l'espérance mathématique $\mathbb{E}[Q(x,\omega)]$ devient la somme $\sum_{\omega\in\Omega} \pi_\omega[Q(x,\omega)]$. Dans ce cadre, il est donc évident que la fonction $f(x) = \varphi(x) + \mathbb{E}[Q(x,\omega)]$ est non seulement *non lisse* (car le sous-problème (0.0.7) peut avoir plusieurs solutions), mais aussi *difficile à évaluer* : un oracle pour f doit résoudre $|\Omega|$ problèmes de programmation linéaire (0.0.7) pour calculer $f(x)$ et un vecteur $g \in \partial f(x)$. Selon la dimension du sous-problème et le nombre de scénarios $|\Omega|$, l'oracle peut être très coûteux en calcul. Par exemple, l'oracle du problème linéaire stochastique à deux étapes du chapitre 4 résultant du problème de planification du réseau de gaz naturel dans [32], avec seulement $|\Omega| = 200$ scénarios, prend 34 minutes pour calculer $f(x)$ et $g \in \partial f(x)$ sur un ordinateur relativement puissant[1].

Si les temps de calcul sont prohibitifs, comment pouvons-nous résoudre les problèmes d'optimisation de cette nature ? Trois alternatives sont possibles : (i) examiner plus de scénarios et réaliser une optimisation inexacte des sous-problèmes; (ii) effectuer une optimisation exacte des sous-problèmes mais en considérant moins de scénarios sélectionnés par une procédure mathématique assurant la stabilité des résultats; (iii) une combinaison des approches (ii) et (iii).

Le chapitre 4 présente l'article [32] sur un problème énergétique stochastique à deux étapes provenant de la PETROBRAS, la société pétrolière et gazière brésilienne. Le chapitre modélise les incertitudes du problème de la planification et de la gestion à long terme du réseau brésilien de gaz naturel, et applique l'approche (ii) citée ci-dessus combinée aux méthodes de faisceaux et à la décomposition stochastique à deux étapes pour résoudre le problème sous-jacent. Les stratégies étudiées ont été mises en œuvre dans le logiciel `MONGE`, qui sera utilisé en 2019 lors de la renégociation du contrat d'approvisionnement de gaz naturel entre le Brésil et la Bolivie.

Programmation stochastique sous contrainte en probabilité.
En termes généraux, un problème d'optimisation stochastique impliquant une contrainte en probabilité peut être écrit sous la forme de (0.0.1), où $\mathfrak{c}$ dépend d'une mesure de probabilité, e.g.

$$\mathfrak{c}(x) := p - \mathbb{P}[G(x,\omega) \geq 0]\,.$$

Dans cette notation, $G : \mathbb{R}^n \times \Omega \to \mathbb{R}^m$ est une fonction donnée, $\omega : \Omega \to \mathbb{R}^m$ est un vecteur aléatoire m-dimensionnel défini sur un espace de probabilité $(\Omega, \mathcal{A}, \mathbb{P})$, et $p \in (0,1]$ est un paramètre pré-spécifié. La contrainte en probabilité

$$\mathfrak{c}(x) \leq 0 \quad \equiv \quad \mathbb{P}[G(x,\omega) \geq 0] \geq p \tag{0.0.8}$$

signifie que le vecteur de décision $x \in \mathbb{R}^n$ est admissible si, et seulement si, le système d'inégalités aléatoires $G(x,\omega) \geq 0$ est satisfait avec une probabilité d'au moins p [166]. Des contraintes en probabilité sont rencontrées dans de nombreux problèmes d'ingénierie impliquant des données incertaines. Il existe des applications dans les télécommunications, l'expansion des réseaux électriques, la gestion des réservoirs hydroélectriques, etc.

Puisque la fonction $\mathfrak{c}$ ci-dessus est non linéaire, écrire la contrainte sous la forme $\mathfrak{c}(x) \leq 0$ dans (0.0.1) fait que le problème apparaît comme un problème classique de programmation non linéaire. Cependant, cette formulation néglige une difficulté cachée : dans la plupart des cas, les valeurs explicites de $\mathfrak{c}$ ne sont

[1] Intel Xeon X5650 2.67GHz, avec 2 processeurs, 48 Go de mémoire RAM, 64 bit Windows Server 2008.

pas disponibles. De plus, les calculs sont souvent inexacts, car calculer la probabilité $\mathbb{P}[G(x,\omega) \geq 0]$ (pour un point x donné) implique généralement de résoudre une intégration numérique multidimensionnelle et/ou d'appliquer des méthodes de (quasi) Monte Carlo. Quelques difficultés supplémentaires sont liées à la différentiabilité et à la convexité de $\mathfrak{c}$: parfois, cette fonction ne parvient pas à être convexe et différentiable même lorsque $-G$ est convexe et différentiable en x [86, 87].

En général, les modèles sous contraintes en probabilités traitent plus explicitement la distribution de probabilités elle-même, alors que les modèles avec recours représentent le caractère aléatoire par un nombre fini de scénarios utilisés pour approximer l'espérance mathématique par une somme pondérée finie. Pour cette raison, les modèles sous contrainte en probabilités peuvent être mathématiquement plus compliqué que les modèles stochastiques avec recours, car ils explorent mieux les informations contenues dans la distribution de probabilité.

La programmation sous contrainte en probabilité est généralement divisée en différentes classes de problèmes en fonction de leur nature aléatoire (distributions de probabilités discrètes, continues et elliptiques) et des propriétés de G (linéaire, non linéaire, non séparable ou séparable, i.e., $G(x,\omega) = \mathfrak{g}(x) - \omega$, avec $\mathfrak{g} : \mathbb{R}^n \to \mathbb{R}^m$).

En considérant la classe des contraintes en probabilités dont le vecteur de décision et le vecteur aléatoire sont séparés, i.e., $\mathbb{P}[G(x,\omega) \geq 0 \equiv \mathbb{P}[\omega \leq \mathfrak{g}(x)]$, le théorème de Sklar garantit que la fonction de probabilité peut être représentée par une fonction composite impliquant $\mathfrak{g}(x) = (\mathfrak{g}_1(x), \ldots, \mathfrak{g}_m(x))$, les distributions marginales F_i, $i = 1, ..., m$, de ω, et une copula $\mathfrak{C} : [0,1]^m \to [0,1]$:

$$\mathbb{P}[\omega \leq \mathfrak{g}(x)] = \mathfrak{C}(F_1(\mathfrak{g}_1(x)), \ldots, F_m(\mathfrak{g}_m(x)))\,.$$

Une copula est une distribution de probabilité multivariée pour laquelle la distribution de probabilité marginale de chaque variable est uniforme. Le chaptire 5 présente l'article [11] qui examine la convexité éventuelle de l'ensemble admissible

$$X(p) := \{x \in X : \mathfrak{C}(F_1(\mathfrak{g}_1(x)), \ldots, F_m(\mathfrak{g}_m(x))) \geq p\}\,.$$

La convexité éventuelle signifie que $X(p)$ est convexe pour tout p assez grand. C'est une propriété utile en pratique, car on se préoccupe généralement de la convexité de $X(p)$ quand p est proche de 1 (un système stochastique d'inégalités devrait être satisfait avec une probabilité élevée). Dans [11], nous avons montré que la grande classe de Copulæ Archimedeannes donne une convexité éventuelle de l'ensemble $X(p)$ (à condition que certains ensembles de niveaux de $\mathfrak{g}_i$ soient convexes). Naturellement, la convexité éventuelle de $X(p)$ n'implique pas que la fonction $x \mapsto -\mathfrak{C}(F_1(\mathfrak{g}_1(x)), \ldots, F_m(\mathfrak{g}_m(x)))$ soit convexe, mais quasi-convexe (sur l'ensemble admissible). Le travail [11] propose la première méthode de niveau capable de traiter une fonction de contrainte possédant des propriétés de convexité généralisées (par exemple la quasi-convexité). Tous les détails sur ce sujet sont donnés dans le chapitre 5.

Toujours dans le cadre de la fonction de probabilité séparable, si la variable aléatoire est unidimensionnelle (i.e., $m = 1$), alors le problème de la minimisation d'une fonction f sur $X \cap \{x \in \mathbb{R}^n : \mathbb{P}[\omega \leq \mathfrak{g}(x))] \geq p\}$ peut être exprimé dans une *forme déterministe* en employant le concept de p-quantile de la fonction de distribution $F(\mathfrak{g}(x)) = \mathbb{P}[\omega \leq \mathfrak{g}(x)]$:

$$\min_{x \in X} \; f(x) \quad \text{s.t.} \quad \mathfrak{g}(x) \geq F^{-1}(p)\,.$$

Calculer le p-quantile $F^{-1}(p)$ n'est pas une tâche difficile (quand $m = 1$) et peut être effectué par des routines de probabilités et de statistiques pour un large éventail de distributions unidimensionnelles. Une fois que $F^{-1}(p)$ est calculé, le problème ci-dessus devient un problème d'optimisation déterministe résoluble par des algorithmes standard.

Dans le cas multidimensionnel $m > 1$, le concept de p-quantile a été étendu à ce qu'on appelle un *point p-efficient* : *Un vecteur $v \in \mathbb{R}^m$ est appelé un point p-efficient de la distribution de probabilité de ω, si $\mathbb{P}[\omega \leq v] \geq p$ et qu'il n'y a pas de $y \leq v$, $y \neq v$ tel que $\mathbb{P}[\omega \leq y] \geq p$.*
Au lieu d'une seule valeur $F^{-1}(p)$, l'ensemble $\mathcal{V}$ des points p-efficient peut contenir un nombre infini

de points (seulement un nombre fini si le support de la distribution de probabilité est fini). Avec cette définition, le problème (0.0.1) avec $\mathfrak{c}$ donné dans (0.0.8) peut être réécrit comme

$$\min_{x,v} f(x) \quad \text{s.t.} \quad \mathfrak{g}(x) \geq v\,,\ x \in X \ \text{ and } \ v \in \mathcal{V}.$$

Du point de vue algorithmique, le principal avantage de cette formulation est qu'elle exempte du calcul des sous-gradients de la fonction de probabilité. Mais cette propriété intéressante a un prix : pour calculer un point p-efficient, il faut résoudre un sous-problème combinatoire dont la dimension dépend de la taille de l'échantillon généré pour approximer les paramètres incertains. Que faire si, pour accélérer les calculs, nous considérons des points p-efficients approximés ? Quel type de résultats en termes de qualité de solution pouvons-nous espérer ?

Pour répondre à ces questions, dans le chapitre 6, nous presentons l'article [4] qui propose un processus itératif pour calculer des points à partir de $\mathcal{V}$ en résolvant inexactement les sous-problèmes combinatoires. En adoptant un point de vue dual, nous avons développé une solution qui inclut et étend plusieurs formulations existantes. Notre approche, qui peut s'appliquer à la fois aux variables aléatoires discrètes et continues, représente une contribution sur trois fronts. Premièrement, en étendant au cadre primal-dual la théorie sur les méthodes de faisceaux inexactes développées dans [149], nous avons révélé l'impact de l'inexactitude (de la fonction dual) sur le problème primal. Deuxièmement, nous avons conçu de nouvelles approches inexactes appelées *on-demand accuracy* qui ont donné des meilleurs résultats dans nos expériences numériques. Troisièmement, grâce à la vue unificatrice de notre article, nous avons prouvé la convergence d'une généralisation des techniques *Regularized Dual Decomposition* et *Progressive Augmented Lagrangian* de [56, 59]. Le chapitre 6 discute en détail les approches ci-dessus.

Toutes les stratégies brièvement discutées ici sont détaillées, en anglais, dans les chapitres 2, 3, 4, 5 et 6, tandis que le chapitre 1 reprend et développe ce résumé.

Contents

Chapter 1

Introduction

This document highlights specific aspects of my research and gives an overview of my contributions to both practical and theoretical sides of the mathematical programming. Motivated by real-life applications from the industry of energy, the core of my research lies in the algorithmic aspects of nonsmooth optimization and stochastic programming, but has also an intersection with more theoretical subjects such as variational analysis and strong convergence in Hilbert spaces of some bundle methods.

The present chapter is the entry door for this document: Section 1.1 presents an overview of the subfields of my research and Section 1.2 highlights my scientific contributions to convex nonsmooth optimization, mixed-integer programming, DC (difference-of-convex functions) programming, stochastic programs with recourse and chance-constrained programming. Differently from the others chapters, this introduction has a concise and conversational style, avoiding a significant amount of technicalities. The goal is to make the content of this chapter accessible not only to experts on nonsmooth and or stochastic programming but also to a broader audience.

The remaining of the document brings forward five of my articles on nonsmooth optimization and stochastic programming that have appeared in specialized journals of mathematical programming in the years 2016, 2017 and 2018. These materials were select to give a flavor on my various interests, subjects, and developments. Deterministic nonsmooth optimization is considered in Chapters 2 and 3, where the former deals with the convex setting by presenting new extensions of the so-called level bundle methods [50], and the latter investigates a class of algorithms with an inertial force for addressing nonsmooth DC programs [51]. Chapter 4 describes the long-term design and operation planning problem of the Brazilian natural gas network investigated in [32]. Such a critical application is modeled as a two-stage stochastic linear program and solved by combining decomposition, a bundle algorithm, and scenario reduction techniques. Chapters 5 and 6 deal with chance-constrained programs: eventual convexity of Copulæ structured chance-constrained problems [11] are presented in Chapter 5, and Chapter 6 reports the work [4] on optimization techniques based on p-efficient points and inexact bundle methods. Finally, Chapter 7 closes with some remarks and presents some future directions of my research. A list of all my activities as a researcher, in the form of an extended Curriculum Vitæ, is given in the Appendix.

1.1 Mathematical programming: fields of action

Mathematical programming is the branch of mathematics concerned with the theory and methods for finding extrema of functions on sets of given vector spaces. The term "mathematical programming" is related to the fact that the goal of solving a mathematical problem (of finding a maximum or a minimum of a function over a feasible set) is choosing a *program/plan of action*. For instance, an electric power company seeks for a program of action to generate enough electricity to meet a load demand at the minimum cost.

The formulation of the problems of mathematical programming (or simply optimization problems) considered in this document reads as

$$\min_{x \in X} \; f(x) \quad \text{s.t.} \quad \mathfrak{c}(x) \leq 0\,, \tag{1.1.1}$$

where $X \neq \emptyset$ is contained in an open set O of a Hilbert space $\mathcal{H}$ and $f, \mathfrak{c} : O \to \mathbb{R}$ are given nonsmooth functions. In most of this document, $\mathcal{H}$ is simply the n-dimensional Euclidean space $\mathbb{R}^n$. Note that in the nonsmooth setting, there is no loss of generality in considering a scalar constraint function $\mathfrak{c}$, as it can be defined as the maximum of all the constraint functions.

One customarily distinguishes the following branches of mathematical programming: linear, quadratic, convex, nonsmooth, stochastic, (mixed-)integer/discrete, DC (difference-of-convex functions) programming, and others. Naturally, a given optimization problem can fit more than one branch: for instance, a stochastic programming problem can be nonsmooth and convex at the same time. Incidentally, it is in this intersection of mathematical programming branches that a significant part of my research lies [4, 23, 148, 199]. Most of my works deal with nonsmooth convex optimization problems, that is, formulation (1.1.1) with X a convex set and with f and $\mathfrak{c}$ convex objective and constraint functions, e.g. [5, 9, 21, 45, 48, 50, 148, 149]. Some of my publications consider convex mixed-integer programming problems (MINLPs): X is a mixed-integer set [53, 150, 203]. Recently, my research has been extended to a broad class of nonconvex problems in which X is convex, but f and $\mathfrak{c}$ are DC functions [51, 201].

1.1.1 Why nonsmooth optimization?

Nonsmooth programming investigates optimization problems in the form of (1.1.1) whose objective function f or constraint $\mathfrak{c}$ is not continuously differentiable. To solve such problems, it is necessary to consider a special mathematical object: the subdifferential of a convex function f, denoted by $\partial f(x)$, which is a fundamental concept in convex analysis. If f is smooth (i.e., continuously differentiable) at a given point x, then the subdifferential of f at x is a singleton and coincides with the gradient $\nabla f(x)$. Otherwise, in the convex setting, the subdifferential of f at x is a set in $\mathcal{H}$ given by

$$\partial f(x) := \{g \in \mathcal{H} : \; f(y) \geq f(x) + \langle g, y - x\rangle \;\; \forall\, y \in \mathcal{H}\}\,.$$

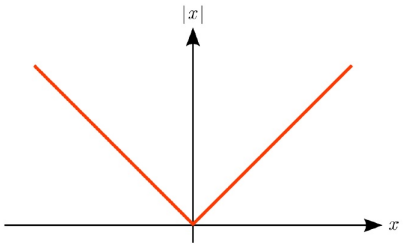


Figure 1.1: Example of a nonsmooth function on $\mathbb{R}$: $f(x) = |x|$. Its subdifferential is given by $\partial f(x) = -1$ if $x < 0$, $\partial f(x) = 1$ if $x > 0$, and $\partial f(0) = [-1,\, 1]$.

This definition says that $\partial f(x)$ is constructed with the slopes of the hyperplanes supporting the epigraph of f at $(x, f(x))$. Each element g of $\partial f(x)$ is called a *subgradient*.

The primary reason for investigating methods for nonsmooth optimization is that nonsmoothness of the functions precludes application of any method for smooth optimization problems. Often in practical situations, (sub)gradients of the involved functions are not computed exactly. In smooth optimization, the gradients are frequently obtained from the function values by finite differences. This approach is valid only in the smooth case, because $f'(x; d) = \langle \nabla f(x), d\rangle$ is linear in d and can be approximated by difference quotients. When f is not differentiable, the mapping $x \to \partial f(x)$ is not continuous. Therefore,

difference quotients do not necessarily belong to the subdifferential, not even in the limit [29, page 125]. In fact, nonsmoothness even hinders the application of derivative-free methods (algorithms which do not require the calculation of derivatives): convergence analysis of such methods strongly depends on differentiability assumptions [42].

Furthermore, the study of nonsmooth programming is motivated by its numerous applications. Indeed, minimization problems of economic origin often lead to functions that are not continuously differentiable. Very often, nonsmoothness is introduced artificially by decomposition techniques employed to tackle real-life optimization problems, which are in general large-scale and heterogeneous in nature. This is the case of the natural gas network planning problem presented in Chapter 4.

1.1.2 Why stochastic programming?

Stochastic programming is the branch of mathematical programming in which one studies the theory and methods for the solution of optimization problems that depend on uncertain parameters having *known probability distributions*. As pointed out in the recent article [160], the assumption on having a known probability distribution has been replaced in some applications (such as portfolio management) by the weaker assumption that only a set of distribution is known which includes the true distribution. Such a weaker assumption yields models that fit better in the field of *robust optimization*, a branch of the mathematical programming that is not covered in this document.

Stochastic optimization is sometimes referred to as *optimization under uncertainty*, but this latter designation should be avoided because is so broad that it encompasses optimization problems for which no stochastic model (depending on a known probability distribution) is available. Due to the presence of random parameters, the theory of stochastic programming combines concepts of optimization and the theory of probability and statistics [181].

The study of stochastic programming is motivated by its numerous applications: stochastic optimization models occur in almost all areas of science and engineering, from power system management, transportation to finance and others. Such models are usually employed as substitutes for deterministic formulations when the latter present themselves as unsatisfactory or inappropriate to tackle decision-making in real-life applications. In general, stochastic optimization models give rise to (very) large-scale optimization problems that can only be solved with the help of decomposition techniques and specialized algorithms. Invariably, techniques such as Benders' decomposition, Lagrangian relaxation, two-stage or multistage decomposition, yield objective functions that are not only *nonsmooth* but also *difficult to evaluate*. These two issues are present in both subareas of the stochastic programming, namely *stochastic programming with recourse* and *chance-constrained programming*, to be considered in Subsection 1.2.2.

In optimization problems with recourse one needs to solve several subproblems to estimate the nonsmooth function f (generally given in a form of expectation), and in chance-constrained problems the probability measure defining $\mathfrak{c}$ is assessed via multidimensional numerical integration and or simulation. In both cases, evaluating exactly the involved functions in (1.1.1) in reasonable CPU time is difficult or impossible in general. These issues have led me to some questions that have shaped my research in the last years.

1.1.3 Some questions that enlighten my research

It's not the answers, but the questions that move the world, said the propaganda of a Brazilian television channel[1]. Naturally, the questions constitute the driving force in the scientific research. Concerning the areas of nonsmooth optimization and stochastic programming, the questions that have influenced my research are related to the design and mathematical analysis of efficient algorithms to tackle nonsmooth optimization problem of the form (1.1.1), whose involved functions and subgradients are difficult to evaluate: Is it possible to design fast computational routines (black-boxes, oracles) to efficiently estimate functional values and subgradients in stochastic programming? If the objective function f or and

[1] Canal Futura.

constraint is/are inexactly evaluated along the optimization process of certain algorithms of the bundle method family [29], can we still have some convergence guarantees? What is the solution quality we can expect, and what are the CPU time savings? These questions have been addressed in the works [48, 133, 149] that show how bundle methods can be modified to efficiently handle several types of inexactness coming from the functions and subgradient evaluations.

For functions whose exact evaluations are possible, although too time-consuming: Is it possible to design optimization algorithms that use inexact information in most of the iterations but still eventually compute an exact solution to the problem? How do the worst-case complexity results of such methods relate to the complexity of their counterpart using exact information? The publications [10, 148] have addressed these questions by considering a special class of bundle methods, the level variants. In particular, the paper [148] introduced a more flexible type of inexact black-boxes called *oracles with on-demand accuracy* that has been used in the mathematical programming community to tackle general nonsmooth programs [12], stochastic programs with recourse [156, 208, 209] and chance-constrained problems [203], [4]. The latter work is presented in Chapter 6.

When dealing with chance-constrained problems, nonsmoothness and inexactness of the probability constraint are not the only difficulties: the lack of nonconvexity makes the problem even more difficult. Are the (upper) level sets of particular classes of probability functions convex for large enough prespecified probability level $p \in (0, 1]$? This question, that is relevant in both theory and practice, is addressed in Chapter 5.

When nonconvexity comes into play in nonsmooth problems of the form (1.1.1), finding a global solution with an optimality certificate is out of reach even for problems of moderate size. Computing stationary points becomes then the only doable goal. To this end, not only appropriate constraint qualifications must be investigated but also implementable forms for computationally checking stationarity. For the broad class of DC problems (1.1.1) whose objective $f = f_1 - f_2$ and constraint $\mathfrak{c} = c_1 - c_2$ are DC functions, is it possible to weaken assumptions on $\mathfrak{c}$ (and point of interest) issuing Slater-like constraint qualifications? Would calmness of $\mathfrak{c}$ suffice as constraint-qualification yielding certain definitions of stationarity? These questions are under investigation, but partial answers can be found in the recent work [201]. Concerning convex-constrained DC programs, is it possible to design algorithms for computing better-quality stationary points without relying on further assumptions on f? This is the subject of Chapter 3.

The scientific research develops from the questions. But where do the questions come from? In my case, they come from the thirst for knowledge, from curiosity, and from real-life optimization problems coming from the industry of energy, more specifically energy management problems.

1.1.4 Energy management: as an application and as a source of inspiration

Energy management is about efficiently using/employing resources for balancing energy demands. It includes the planning and operation of energy production and energy consumption units. As an example, in electrical systems one needs to balance power and demand in real-time, which implies appropriate planning and possibly expansion of power systems. Although cost minimization and income maximization are the most common objectives, conservation of resources and climate protection may gain protagonism as new legislation on the energy system evolution comes in.

In the class of energy management problems, the importance of mathematical programming methods has been recognized a long time. With the increasing share of intermittent renewable sources of power generation, the need of handling uncertainty within such optimization problems has become of paramount importance for the quest of higher reliability of the system and higher profitability. As one can notice, not only the questions move the world as previously quoted, but also the world moves the questions: In this new paradigm, how can we deal with the random nature of renewable sources of power generation? Stochastic programming provides the answers, and for this reason, stochastic optimization methods have become essential tools in the industry of energy.

As well known, real-life energy problems give rise to very large optimization problems that, in general, can only be solved with the help of decomposition techniques. As already mentioned, such techniques lead to objective functions that are not only nonsmooth but also difficult to evaluate. In some applications of interest, exact evaluations of the function (and its subgradient) are prohibitive in terms of computational time. Very often, not only differentiability assumptions are absent, but also convexity. These issues rise optimization problems possessing a high level of difficulty, which is sometimes prohibitive in the large-scale setting. Therefore, new techniques from variational analysis and computational mathematics are required to improve solution methods in this setting. To put it in a nutshell, energy management problems constitute a source of inspiration and a challenging application of mathematical programming tools.

1.2 Scientific achievements: a bird's eye view

This section presents an overview of my research on nonsmooth optimization and stochastic programming and highlights my contributions to these fields. The section starts with a discussion on algorithms for nonsmooth optimization by presenting bundle methods, DC programming, to stochastic programs with chance constraint.

1.2.1 Nonsmooth optimization

As already mentioned, the nondifferentiability of the involved functions precludes application of any method for smooth optimization problems, including those which do not require the calculation of derivatives. Nonsmooth optimization problems require special methods, taking into account the discontinuity of the subdifferential. Among such special techniques, the subgradient and the cutting-plane methods are perhaps the most well-known approaches for dealing with nonsmooth optimization problems. Such methods are also famed for having slow convergence (in some cases, the number of trial points generated by the cutting-plane method grows exponentially with the problem dimension, [142, pp. 158-160]). When accuracy in the solution and reliability are a concern, bundle methods [29] are the algorithms of choice for nonsmooth convex optimization. Is it also the case when the involved functions can only be assessed through an oracle providing inexact information? What if either the feasible set or the objective function lacks convexity? Are bundle methods still the "algorithms of choice" in these situations? These questions are addressed in the sequel.

1.2.1.1 Convex bundle methods

Let us, for the moment, focus on problem (1.1.1) without the nonlinear constraint $\mathfrak{c}$ and make the assumption that f is a convex function. Moreover, we assume that an (exact) oracle for f is available: for any given point $x \in X$, the oracle provides us with $f(x)$ and a subgradient $g \in \partial f(x)$.

Having collected a *bundle* of information $\{(f(x_1), g_1), \ldots, (f(x_k), g_k)\}$ on the trial points x_j, $j \in J_k := \{1, \ldots, k\}$, convexity of f ensures that the cutting-plane model

$$\check{f}_k(x) := \max_{j \in J_k} \{f(x_j) + \langle g_j, x - x_j \rangle\} \quad \text{satisfies} \quad \check{f}_k(x) \leq f(x) \quad \text{for all } x \in \mathcal{H}.$$

With such a model at hands, the cutting-plane method of [103] defines the new trial point as a solution of the master program

$$x_{k+1} \in \arg\min_{x \in X} \ \check{f}_k(x)\,, \tag{1.2.1}$$

which is a linear programming problem if X is polyhedral. One main disadvantage of the method becomes evident: since the bundle of information indexed by elements in J_k grows with the iterative process, subproblem (1.2.1) becomes more and more difficult to solve. Differently from the Kelley's cutting-plane method, most bundle methods have limited memory: the bundle of oracle information

can be kept bounded saving computational memory without impairing convergence. This is particularly interesting for large-scale optimization problems. Without making any notational distinction between the full cutting-plane model issued with $J_k = \{1, \ldots, k\}$ and a possibly more economical one given by a certain subset $J_k \subset \{1, \ldots, k\}$, we now recall the main elements of bundle methods.

Standard bundle methods make use of three main ingredients: a convex model $\check{f}_k$ that (ideally) approximates the function f from below; a stability center $\hat{x}_k$ that is in general the "best" point generated by the iterative process; and a bundle parameter ($t_k > 0$, $f_k^{\text{lev}} \in \mathbb{R}$ or $\rho_k > 0$) to be updated at each iteration k. A new iterate x_{k+1} of a bundle method depends on these ingredients, whose organization defines different variants:

Proximal bundle variant

$$x_{k+1} := \arg\min_{x \in X} \ \check{f}_k(x) + \frac{1}{2t_k} \|x - \hat{x}_k\|^2 \tag{1.2.2}$$

Level bundle variant

$$x_{k+1} := \arg\min_{x \in X} \ \frac{1}{2} \|x - \hat{x}_k\|^2 \quad \text{s.t.} \quad \check{f}_k(x) \leq f_k^{\text{lev}} \tag{1.2.3}$$

Trust-region bundle variant

$$x_{k+1} := \arg\min_{x \in X} \ \check{f}_k(x) \quad \text{s.t.} \quad \|x - \hat{x}_k\|^2 \leq \rho_k. \tag{1.2.4}$$

It is known that for properly chosen prox, level and trust-parameters (t_k, f_k^{lev}, ρ_k) and the same stability center $\hat{x}_k$, it is always possible to generate the same next iterate by solving either (1.2.2), (1.2.3) or (1.2.4). In this theoretical sense the three approaches can be considered equivalent. However, details of the implementation and practical performance can be quite different: because the parameters are updated by strategies specific to each of the methods and the corresponding rules are not related in any direct way.

The last few years have seen the advent of a new generation of bundle methods, capable of handling inexact oracles, polluted by "noise": given $x \in \mathbb{R}^n$, an inexact oracle returns with

$$\left\{ \begin{array}{l} f_x = f(x) - \eta_x^v \ \text{ and} \\ g_x \in \mathbb{R}^n \ \text{ such that } \ f(\cdot) \geq f_x + \langle g_x, \cdot - x \rangle - \eta_x^s \\ \text{with } \eta_x^v \leq \eta \text{ and } \eta_x^s \leq \eta \text{ for all } x \in \mathbb{R}^n. \end{array} \right. \tag{1.2.5}$$

In the above scheme, the bound $\eta \geq 0$ is possibly unknown. The subscripts v and s on the errors in (1.2.5) make the distinction between function *value* and *subgradient* errors. Notice that if $\eta = 0$, then (1.2.5) boils down to an exact oracle. If $\eta > 0$ but $\eta_x^s = 0$ (i.e., the linearization issued by f_x and g_x lies under the function f), then (1.2.5) is a *lower oracle* in the parlance of [148]. Instead, if η_x^s can be positive we have an *upper oracle*: the value f_x can overestimate $f(x)$. We care to mention that with inexact oracles of the upper type, the inexact model

$$\check{f}_k(x) = \max_{j \in J_k} \{f_{x_j} + \langle g_{x_j}, x - x_j \rangle\}$$

may cut off the graph of the function f. This undesirable feature complicates convergence analysis of bundle methods, and generally requires an extra procedure to deal with errors.

Many applications of optimization to real-life problems lead to nonsmooth objective and or constraint functions that are assessed through inexact oracles of the above form. For example, this is the typical case in Lagrangian relaxation of large-scale (possibly mixed-integer) optimization problems, in stochastic programming (as it will be discussed in the next section), and in robust optimization, where the oracles perform some numerical procedure to evaluate functions and subgradients, such as solving one or more optimization subproblems, multidimensional integration, or simulation.

Proving convergence of a bundle method is never simple and coping with inexact oracles substantially increases the technicalities. Besides, several variants exist to deal with noise, each one needing an ad hoc

proof to show convergence. In the joint work with Claudia Sagastizábal (Unicamp, Brazil) and Claude Lemaréchal (INRIA, France) [149] we provided a synthetic convergence theory, in which we highlight the main arguments and specify which assumption is used to establish each intermediate result. The framework is comprehensive and generalizes in various ways a number of algorithms proposed in the literature. Our depth analysis covers many proximal bundle methods in the literature and opened the way to more variants [4, 12].

A flexible class of *lower oracles with on-demand accuracy*, motived by stochastic programs with recourse, was introduced in [148] and further investigated in [9, 203]. An oracle of this class satisfies (1.2.5) with the following additional assumptions, where $0 \leq \eta_k \leq \eta$ and $f_k^{tar} \in \mathbb{R}$ are given parameters:

$$\eta_x^s \equiv 0 \quad \text{and} \quad \eta_x^v \leq \eta_k \text{ if } f_x \leq f_k^{tar}. \tag{1.2.6}$$

This kind of oracles provide information with accuracy up to η_k for those points whose function approximations do not exceed the given target f_k^{tar} (see Figure 1.2.1.1). The target value f_k^{tar} can be, for

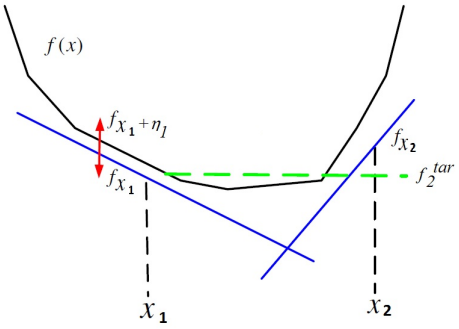


Figure 1.2: Oracle with on-demand accuracy. When the oracle identifies that the approximate value f_{x_2} is greater than the given target f_2^{tar}, then it returns with the poor information (f_{x_2}, g_{x_2}): there is no need to spend CPU time for computing exactly $f(x_2)$ once x_2 is identified to be no better than the previous iterate x_1. Notwithstanding, the cheap and new oracle information (f_{x_2}, g_{x_2}) is possibly of poor quality, but can be employed to improve the cutting-plane model of f. If the oracle is of the lower type, then the cutting-plane model never cuts off the function.

instance, the best known upper bound for the optimal value of the problem: $f_k^{tar} = \min_{j \leq k}\{f_{x_j} + \eta_{x_j}^v\}$. Other possibilities are given in [148]. The concept of oracles with on-demand accuracy has been extend in [12] where the authors consider oracles providing not only lower but also upper estimates for the function. As just mentioned, dealing with inexact oracles is not a straightforward task. The situation is more involving when the goal is to compute an exact solution to the problem by making use of inexact (but controlled) information: the two parameters f_k^{tar} and η_k must be carefully handled. The level bundle method algorithms proposed in [148], with Claudia Sagastizábal, eventually find an exact solution to the problem by automatically setting up these two additional parameters and ensuring $\lim_{k\to\infty} \eta_k = 0$. We proved mathematically that, although the new method uses inexact oracle information, not only an exact solution to the problem is eventually found but also the same complexity result $\mathcal{O}(\frac{1}{\epsilon^2})$ of exact level bundle methods is preserved. (The worst-case complexity of the inexact methods is the same as the one of exact algorithms except for a multiplicative constant). The paper [148] was awarded the 2014 Charles Broyden Prize[2].

While the level bundle methods of [148] deal with inexact oracles whose errors can be controlled, the work [133], in collaboration with Jérôme Malick (CNRS, France) and Sofia Zaourar (NAVER LABS, France), proposes a new inexact level bundle algorithm for dealing with very general inexact lower oracles. We proposed an original implicit procedure to handle excessive inexactness permitting asymptotic convergence of the algorithm without boundedness assumption.

[2] http://explore.tandfonline.com/page/est/charles-broyden-prize

The work [149] deals only with proximal bundle methods, whereas [133, 148] investigate the level variants. Overall, there seems to be a consensus that for solving unconstrained problems proximal bundle methods are very good choices, although the updating rule for t_k is somewhat of an issue (at least from the viewpoint of combining theory and efficiency). On the other hand, there is some evidence that for constrained problems level bundle methods might be preferable. Also, strategies for updating the level parameter f_k^{lev} are readily available. It is thus appealing to try to combine the attractive features of both approaches in a single algorithm that performs for unconstrained (respectively, constrained) problems as well as proximal bundle methods (respectively, level bundle methods), or maybe even better in some cases. To this end, with Mikhail Solodov (IMPA, Brazil) we proposed what we call a *doubly stabilized bundle method*, that combines both proximal and level stabilizations in the same subproblem, namely

$$x_{k+1} := \arg\min_{x\in X} \ \check{f}_k(x) + \frac{1}{2t_k}\|x-\hat{x}_k\|^2 \quad \text{s.t.} \quad \check{f}_k(x) \le f_k^{\text{lev}}. \tag{1.2.7}$$

As showed in [48], the unique solution to problem (1.2.7) is also a solution to at least one of the problems (1.2.2) or (1.2.3). Hence, the method indeed combines the proximal and the level approaches, "automatically" choosing between the two at every step: each iteration of the doubly stabilized method is either a level iteration (issued by subproblem (1.2.3)) or proximal iteration (issued by subproblem (1.2.2)); see Figure 1.3. Another interesting feature of the doubly stabilized method is that it does not

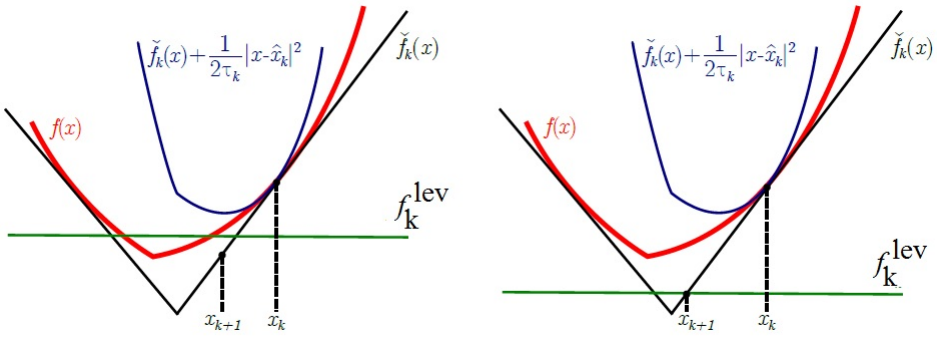


Figure 1.3: Illustration of a proximal iteration (left) and a level iteration of the doubly stabilized bundle method.

require any extra procedures to handle inexactness from the oracle, in contrast to all previous bundle methods in the literature. The doubly stabilized method firstly appeared in the conference paper [49], but it was fully investigated in [48].

From what we have discussed so far, we already have an answer to one of the questions raised at the beginning of this subsection: yes, bundle methods are efficient and robust for dealing with noisy functions. They can be considered as the "methods of choice" even when the involved functions can only be assessed through an oracle providing inexact information.

Without exploiting inexact oracles, other bundle variants are given in [153] and [50]. In particular, the latter article is reported in Chapter 2. As we will see, [50] provides a convergence analysis and presents some extentions of the target radius method briefly introduced at the end of [153]. One of the given extensions is dedicate to solve problems in the form of (1.1.1) with constraint $\mathfrak{c}$ defined by a continuous differentiable and strongly convex function $\mathfrak{s} : X \to \mathbb{R}$:

$$\mathfrak{c}(x) \le 0 \quad \equiv \quad \mathfrak{s}(x) \le \delta\,.$$

Every iterate of the proposed target radius methods is given by

$$x_{k+1} := \arg\min_{x\in X} \ \mathfrak{s}(x) \quad \text{s.t.} \quad f(x_j) + \langle g_j, x - x_j\rangle \le f_{k,j}^{\text{lev}} \quad \forall\, j \in J_k\,. \tag{1.2.8}$$

If the given trial point does not satisfies the nonlinear constraint $\mathfrak{s}(x) \le \delta$, then x_{k+1} is disregarded, $\min_{j\in J_k}\{f_{k,j}^{\text{lev}}\}$ is a valid lower bound on the optimal value of (1.1.1), and new level parameters $f_{k+1,j}^{\text{lev}} >$

$f_{k,j}^{\text{lev}}$ are set up. This is an original and practical rule to define lower bounds for problem (1.1.1). The above formulation makes clear the link between the target radius and level bundle algorithms: take $f_{k,j}^{\text{lev}} := f_k^{\text{lev}}$ for all $j \in J_k$, $\mathfrak{s}(x) := \frac{1}{2}\|x - \hat{x}\|^2$ and compare with (1.2.3). The interest of having individual level parameters (a parameter per linearization) instead of a single level parameter for the model is justified by the "slops" g_j: if the slop yielded by g_j is steeper than the one issued by g_i, then it might be convenient to take $f_{k,j}^{\text{lev}} < f_{k,i}^{\text{lev}}$ in order to try to push x_{k+1} near to the solution set or, in other case, to push it out from the feasible set (i.e. $\mathfrak{s}(x_{k+1}) > \delta$) issuing for free a new and better lower bound estimate. The work [50] proves that the given algorithm possesses dimension-independent iteration complexity and optimal (in the large-scale case) rate-of-convergence results for the minimization of a convex function over a Euclidean ball, a standard simplex, and other domains. An innovative mechanism allowing to keep the method's memory bounded is also proposed. In addition, [50] presents the first level-like bundle method for the particular class of bilevel optimization

$$\min_{x \in \mathbb{R}^n} \ \mathfrak{s}(x) \quad \text{s.t.} \quad x \in \arg\min_{y \in X} f(y)\,.$$

The target radius method and its variants are considered in details in Chapter 2.

Moving now to optimization problems in Hilbert spaces, the two works [5, 21] investigate strong converge of bundle methods. A key procedure in proximal bundle methods for convex minimization problems is the definition of stability centers $\hat{x}_k$, which are points generated by the iterative process that successfully decrease the objective function. In general, a rule to define stability centers depends on the decrease of the function in the following manner, for a given $\kappa \in (0, 1)$:

$$\textbf{If} \quad f(x_{k+1}) \leq f(\hat{x}_k) - \kappa[f(\hat{x}_k) - \check{f}_k(x_{k+1})], \quad \textbf{then} \quad \hat{x}_{k+1} := x_{k+1} \quad \textbf{else} \quad \hat{x}_{k+1} := \hat{x}_k\,.$$

It follows that $f(\hat{x}_k) - \check{f}_k(x_{k+1}) \geq 0$ due to convexity and subproblem (1.2.2). Therefore, the above is indeed a descent-test. In [5] we proposed and investigated a different stability-center classification rule for proximal bundle methods:

$$\textbf{If} \quad f(x_{k+1}) \leq f(\hat{x}_k) - \kappa[f(\hat{x}_k) - \check{f}_k(x_{k+1})], \quad \textbf{then} \quad \hat{x}_{k+1} := \texttt{Proj}_{\mathbb{X}_k}(x_0) \quad \textbf{else} \quad \hat{x}_{k+1} := \hat{x}_k\,,$$

where $\texttt{Proj}_{\mathbb{X}_k}(x_0)$ stands for the convex projection of the initial point x_0 onto a certain level set $\mathbb{X}_k$ of f; see [5, Eq. (17)]. We showed that the proposed bundle variant has at least two particularly interesting features: (i) the sequence of stability centers generated by the method converges strongly to the solution that *lies closest* to the initial point, and (ii) if the sequence of stability centers is finite, $\hat{x}$ being its last element, then the sequence of nonstability centers converges strongly to $\hat{x}$. Property (i) is useful in some practical applications in which a minimal norm solution is required. The results of [5] extends and strengths the convergence results from the seminal work [43] (e.g., the latter work does not prove either (ii) nor that the method provides a solution that is closest to the initial point).

In the other publication [21] with Yunier Bello-Cruz (Northern Illinois University, USA), we not only proved convergence of a level bundle variant but also provided an original manner to update the level parameter f_k^{lev}. This idea has been employed by other colleagues in [38] to prove worst-complexity results and optimal rate-of-convergence for minimizing a convex function over the entire space. The work [21] also exploits a link between level bundle methods and projected subgradient methods. This subject was further studied in [45] by focusing on strong convergence of the latter class of algorithms.

So far, the mentioned bundle methods were designed for convex optimization. The nonconvex setting is discussed in the sequel.

1.2.1.2 Mixed-integer optimization

Many real-life optimization problems are modeled in a mixed-integer setting, involving discrete and continuous decision variables. Optimization algorithms for solving *mixed-integer nonlinear programming* (MINLP) problems have become an important focus of research over the last years. Most of the algorithms for solving problems of this type require the objective and constraint functions to be convex

and differentiable. The latter hypothesis is very often absent in optimization problems coming from the industry of energy, hindering thus the applicability of many powerful algorithms for MINLP. It is thus appealing to investigating optimization methods for *nonsmooth convex MINLP*, i.e., problems as in (1.1.1) where X is a mixed-integer set and f and $\mathfrak{c}$ are nonsmooth convex functions. The two main obstacles in this class of problems are the discrete nature of (a part of) X and nonsmoothness of f and $\mathfrak{c}$. These two obstacles have been extensively addressed individually in the communities of mixed-integer programming and nonsmooth optimization, respectively. However, the combination of these two optimization areas is sparse, although most of the methods for MINLP and methods for nonsmooth programming have as ancestor the same cutting-plane method of Kelley [103].

The work [150] proposes a class of level bundle methods for convex MINLP. In contrast to most methods found in the literature, the proposed approaches do not require the involved functions to be either differentiable or easy to evaluate. Convergence analysis is presented by assuming that each iterate is arbitrarily chosen in a certain localizer set. Although solving mixed-integer linear master programs is optional, numerical results show that when certain regularized master problems are solved along iterations the number of function evaluations is significantly reduced. This is a feature of practical importance, for instance to handle MINLP with a (hard-to-evaluate) probabilistic constraint. Indeed, the ideas of [150] were further explored in [203] to handle, in particular, chance-constrained programming with finite support.

The methods proposed in [150] and [203] are of the (extended) cutting-plane method type: at every iteration the method adds new linearizations (cuts) of the involved functions to the master program

$$x_{k+1} \in \arg\min_{x\in X} \ \frac{1}{2}\,\|x-\hat{x}_k\|_\diamond \quad \text{s.t.} \quad \check{\mathfrak{c}}_k(x)\le 0,\ \check{f}_k(x)\le f_k^{\text{lev}},\ x:=(x_c,x_d)\in\mathbb{R}^{n_c}\times\mathbb{Z}^{n_d},$$

where $\check{\mathfrak{c}}_k$ is a cutting-plane model for $\mathfrak{c}$, and $\|\cdot\|_\diamond$ is an arbitrary norm (in general the ℓ_1 or ℓ_∞ norms yielding a mixed-integer linear master program), and x_c and x_d are the continuous and discrete components of the vector x.

It is well-known that better ("deep") cuts can be obtained by the so-called *outer-approximation* (OA) methods. Algorithms of this class compute trial points by (inexactly) solving the master program

$$(\tilde{x}_c,x_d^{k+1})\in\arg\min_{x\in X}\ \check{f}_k(x) \quad \text{s.t.} \quad \check{\mathfrak{c}}_k(x)\le 0,\ x:=(x_c,x_d)\in\mathbb{R}^{n_c}\times\mathbb{Z}^{n_d},$$

and then the integer variables are fixed and a resulting convex nonlinear subproblem is exactly solved in the continuous variables

$$x_c^{k+1}\in\arg\min_{x\in\mathbb{R}^n}\ f(x_c,x_d^{k+1}) \quad \text{s.t.} \quad \mathfrak{c}(x_c,x_d^{k+1})\le 0,\ (x_c,x_d^{k+1})\in X.$$

The outer-approximation algorithm for MINLP was proposed 1986 in the pioneering work [63]. In this latter paper the authors not only assume the functions to be smooth but also a linear dependence on the integer variables: $f(x)=\varphi(x_c)+q^\top x_d$, where φ is a smooth convex function and q is a given vector. The linearity assumption was removed in 1994 in the seminal paper [68], but the hypothesis on smoothness was still required. Many improvements on the OA algorithm have been proposed along the years, but it was only in 2018 that the assumption on smootheness was completely removed: in the paper [53] with my former PhD student Adriano Delfino (UFTPR, Brazil) we succeeded in proposing outer-approximation algorithms for general nonsmooth convex MINLP problems. To ensure convergence of the methods we needed not only to solve the convex nonlinear subproblem at every iteration (as all OA algorithms do) but also to compute specific subgradients satisfying its KKT system. The latter is a nontrivial task and therefore cannot be expected to be accomplished by an ordinary oracle. By designing a specialized proximal bundle method we managed to compute subgradients satisfying the required conditions and, therefore, any OA algorithm that is convergent for smooth problems is also convergent in the nonsmooth setting when equipped with such a bundle algorithm. This was the main theoretical contribution of [53], which also deals with a chance-constrained MINLP problem. We highlight that the theory of bundle methods studied in [53] gave then a definitive answer to an open question (of more than 30 years) from the MINLP community.

Sometimes, the lack of convexity is not related to the feasible set, but the functions themselves.

1.2.1.3 DC (Difference-of-Convex functions) programming

DC programming forms an important subfield of nonconvex programming and has been receiving much attention from the mathematical programming community. Problems of this class fit formulation (1.1.1) with X a convex set, $f = f_1 - f_2$ and $\mathfrak{c} = c_1 - c_2$, where f_1, f_2, c_1 and c_2 are given convex functions

$$\min_{x \in X^c} f_1(x) - f_2(x) \quad \text{with} \quad X^c := \{x \in X : \, c_1(x) - c_2(x) \leq 0\}\,. \tag{1.2.9}$$

Some applications of (1.2.9) include chance-constrained problems, energy management problems, production-transportation planning problems, cluster analysis and others.

The main advantages of DC programming is that it is an extension of convex programming that is vast enough to cover almost all nonconvex optimization problems (e.g., all lower-C^2 functions are DC), but still allows the use of powerful tools from convex analysis and convex optimization.

As for nonconvex nonsmooth optimization, in nonsmooth DC programs many definitions of stationary points exist too. Which is the strongest one? Can we compute it? To answer the first question we rely on [155], which proves that the sharpest definition is the B(ouligand)-stationary: a feasible point $\bar{x}$ is called a B-stationary point of (1.2.9) if the directional derivative of f with respect to d is nonnegative for all $d \in \mathcal{T}_{X^c}(\bar{x})$, the Bouligand tangent cone of X^c at point $\bar{x} \in X^c$. This is equivalent to

$$f_1'(\bar{x}; d) \geq f_2'(\bar{x}; d) \quad \forall\, d \in \mathcal{T}_{X^c}(\bar{x})\,. \tag{1.2.10}$$

Here $\mathcal{T}_{X^c}(\bar{x})$ is the Bouligand tangent cone of X^c at point $\bar{x} \in X^c$. In the general setting where f_2 (or c_2) is a nonsmooth function, checking the B-stationarity of a given point is a difficult task.

In a joint work with Wim van Ackooij (EDF, France), we investigated in [201] DC-constrained DC programs as in (1.2.9) and proposed a weaker constraint qualification permitting to characterize the Bouligand tangent cone (and therefore the definition (1.2.10)) in a workable form. In addition, we proposed and analyzed (less computational demanding) minimization algorithms to address the problem of computing (under more restrictive hypotheses) B-stationary points to (1.2.9). The proposed algorithms were numerically assessed on a DC reformulation of an energy management problem considering a smart-grid controlled by a local actor (follower) and its interaction with a global actor (leader) in the power system.

When there is no DC constraint, the B-stationarity definition is better known as d-stationarity

$$\partial f_2(\bar{x}) \subset \partial f_1(\bar{x}) + N_X(\bar{x})\,,$$

where $N_X(\bar{x})$ is the normal cone to the convex set X at point $\bar{x}$. As in the DC-constrained setting, checking B-stationarity is not an easy task when f_2 is a general nonsmooth convex function. We therefore shall be satisfied with a weaker stationarity definition, called *criticality*:

$$\emptyset \neq \partial f_2(\bar{x}) \cap \partial f_1(\bar{x}) + N_X(\bar{x})\,.$$

(Notice that criticality and d-stationarity coincide when f_2 is smooth.)

In these convex-constrained setting (i.e., problem(1.2.9) without the nonlinear constraint), the joint paper with Michel Tcheou (UERJ, Brazil) [51] proposes a new family of non-monotone DC algorithms. To the best of acknowledgment, all the DC algorithms in the literature are monotone and, therefore, are easily trapped by poor quality critical points. With the aim of computing critical points that are also d-stationary (without any additional assumption on either f_1 or f_2) we proposed an algorithmic scheme equipped with an inertial-force procedure akin to the Heavy-Ball method of Polyak. In contrast to classical DC algorithm of [193], our approaches do not require solving convex subproblems up to optimality to define trial points: only inexact solutions of the following convex subproblem suffices

$$x_{k+1} \in \arg\min_{x \in X} f_1(x) - \langle g_2^k + \gamma(x_k - x_{k-1}), x\rangle, \quad \text{with given} \quad g_2^k \in \partial f_2(x_k) \;\text{ and }\; \gamma \geq 0.$$

The approach was numerically assessed on large-scale nonconvex and nonsmooth image denoising models, which have become important tools in computer vision systems. More details in Chapter 3.

The manuscript [47] also deals with convex-constraiend DC programs and extends the bundle method of [101] by (i) considering convex-constrained nonsmooth DC programs, (ii) by proving convergence of the resulting proximal bundle algorithm with weaker assumptions on the management of the oracle information (there is no need to keep an individual bundle for f_2), and (iii) by proposing the first implementable algorithm able to compute d-stationary points under the assumption that f_2 is the pointwise maximum of finitely many smooth convex functions. The master program of the bundle method of [47] reads as

$$x_{k+1} \in \arg\min_{x \in X} \ \check{f}_1^k(x) - \langle g_2^k, x\rangle + \frac{1}{t_k} D(x, \hat{x}_k),$$

where $\check{f}_1^k$ is a cutting plane of f_1, D is a Bregman function (e.g. the Euclidean distance) and $\hat{x}_k$ is a given stability center. Instead, the master problem of [101] for unconstrained problems uses a DC cutting-plane model

$$x_{k+1} \in \arg\min_{x \in \mathbb{R}^n} \ \check{f}_1^k(x) - \check{f}_2^k(x) + \frac{1}{2t_k} \|x - \hat{x}_k\|^2,$$

which can be significantly more difficult to solve depending on the number of linearizations used to define the model $\check{f}_2^k$.

All the methods discussed above find applications in stochastic programming.

1.2.2 Stochastic programming

This subsection presents an overview of my research on stochastic programming. The presentation starts with a discussion on how to design oracles providing exact or inexact information on the objective function and subgradient of two-stage stochastic programs with recourse, passing by scenario reduction and regularization techniques in the multistage setting, to the discussion on eventual convexity of chance-constrained programs, where optimization techniques for both chance-constrained problems with finite support and with continuous probability distributions are considered.

1.2.2.1 Stochastic programming with recourse

In stochastic programming with recourse the objective function in (1.1.1) is in general given by

$$f(x) = \varphi(x) + \mathcal{R}[Q(x, \omega)],$$

where ω is a random vector following a known probability distribution, $\varphi : \mathbb{R}^n \to \mathbb{R}$ is a deterministic function, $f : \mathbb{R}^n \times \Omega \to \mathbb{R}$ is a function depending on both the decision variable and random vector, and $\mathcal{R}$ is a risk measure [161],[181, Chapter 6]. In the risk-neutral setting, $\mathcal{R}$ is the expected value operator w.r.t. the probability measure of ω: $\mathcal{R}[Q(x, \omega)] = \mathbb{E}[Q(x, \omega)]$. A risk-averse formulation can be obtained, for instance, with $\mathcal{R}[Q(x, \omega)] = \mathbb{E}[Q(x, \omega)] + r\mathbb{V}ar[Q(x, \omega)]$, with $r > 0$ a given constant and $\mathbb{V}ar$ the variance of $f(x, \omega)$. Other risk measures can be found in [161]. For simplicity, in what follows only the risk-neutral setting is considered to discuss two-stage and multistage stochastic linear programs.

What does recourse mean in stochastic programs?
In a first stage, a decision x is chosen to be feasible with respect to the deterministic first stage constraints: $\{x \in X : \mathfrak{c}(x) \leq 0\}$. Later on, after the realization of the random vector ω a possible deficiency in some second stage constraints has to be compensated by an appropriate recourse decision $y(\omega)$. This process continues until the last stage of decision. Generally speaking, if one makes a *here-and-now* decision x that presents itself wrong when the future event ω reveals, there exists the opportunity to use or do something to compensate such an incorrect decision. This is what *recourse* means.

Two-stage stochastic programs
In a two-stage (linear) programming setting, the recourse function $Q(x, \omega)$ is the optimal value of the

linear programming problem

$$Q(x,\omega) := \begin{cases} \min & \langle q(\omega), y\rangle \\ \text{s.t.} & y \geqslant 0\,, Wy = b(\omega) - T(\omega)x \end{cases}$$

for given vectors $q(\omega)$, $b(\omega)$ and matrices $T(\omega)$ and W. The evaluation of the recourse function can be done by solving the dual linear program

$$\begin{cases} \max & \langle b(\omega) - T(\omega)x, u\rangle \\ \text{s.t.} & W^\top u \leq q(\omega)\,. \end{cases} \tag{1.2.11}$$

Let $u_{x,\omega}$ be a solution of the above subproblem. It is well known that, for all $x \in \mathcal{D}\mathrm{om}(f)$,

$$g := \nabla\varphi(x) - \mathbb{E}[T(\omega)^\top u_{x,\omega}] \quad \in \quad \partial f(x)\,.$$

Evaluating exactly the objective function f and a subgradient g, when the probability distribution of ω is continuous, is impossible in problems of practical relevance. Hence, stochastic optimization models generally consider a scenario formulation: the random vector ω is approximated by a sample of scenarios $\Omega := \{\omega^1, \omega^2, \ldots, \omega^N\}$ with associated probabilities $\pi_\omega > 0$, $\omega \in \Omega$, and the expectation $\mathbb{E}[Q(x,\omega)]$ becomes the sum $\sum_{\omega\in\Omega} \pi_\omega[Q(x,\omega)]$. In this setting, it is thus evident that the function $f(x) = \varphi(x) + \mathbb{E}[Q(x,\omega)]$ is not only *nonsmooth* (because subproblem (1.2.11) can have more than one solution), but also *difficult to evaluate*: an oracle for f must solve $|\Omega|$ linear programming problems (1.2.11) to compute $f(x)$ and a vector $g \in \partial f(x)$. Depending on the subproblem's dimension and on the number of scenarios $|\Omega|$, the oracle can be too time consuming. For instance, the oracle for the two-stage stochastic linear problem resulting from the natural gas planning problem of [32], with only $|\Omega| = 200$ scenarios, takes 34 minutes for computing $f(x)$ and $g \in \partial f(x)$ in a reasonably power computer[3].

If CPU time is an issue, how can we tackle optimization problems of this nature? Is it taking fewer scenarios the only doable way to go? By the Law of Large Numbers, one should look at as much scenarios as possible to estimate well the *true* objective function based on the expectation involving the continuous probability of ω. Hence, if one randomly generates a small sample Ω of scenarios, the sample average approximation of the expectation (and therefore the problem's approximation) can be very poor. Three alternatives to overcome this difficulty are possible: (i) looking at more scenarios and performing inexact subproblem optimization; (ii) performing exact subproblem optimization but considering fewer scenarios selected by some mathematical procedure ensuring stability of results; (iii) a combination of the approaches (ii) and (iii).

By employing approach (i) to speed up calculations, if instead of performing the maximization in (1.2.11) for the considered ω we just take a feasible point $u_{x,\omega}$ (satisfying $W^\top u_{x,\omega} \leqslant q(\omega)$), then an oracle giving

$$f_x := \varphi(x) + \mathbb{E}[\langle b(\omega) - T(\omega), u_{x,\omega}\rangle] \quad \text{and} \quad g_x := \nabla\varphi(x) - \mathbb{E}[T(\omega)^\top u_{x,\omega}]$$

is of *lower type*, in the parlance of [148]: f_x and g_x satisfy (1.2.5) with $\eta_x^s = 0$, i.e.,

$$\begin{cases} f_x = f(x) - \eta_x^v \ \text{ and} \\ g_x \in \mathbb{R}^n \ \text{ such that } \ f(\cdot) \geq f_x + \langle g_x, \cdot - x\rangle \\ \text{with } \eta_x^v \leq \eta \text{ for all } x \in \mathbb{R}^n. \end{cases}$$

If the approximate dual solution $u_{x,\omega}$ is not feasible to (1.2.11), then η_x^s can be positive and the inexact oracle (1.2.5) is of the *upper type*: the value f_x can overestimate $f(x)$.

If ones selects a (much smaller) subset $\Omega_k \subset \Omega$ of scenarios to perform the subproblem optimization, another upper oracle can be obtained by setting f_x and g_x as above but with $\mathbb{E}$ replaced by $\mathbb{E}_k$, the

[3] Intel Xeon X5650 2.67GHz, with 2 processors, 48 Gbyte of RAM Memory, running 64 bit Windows Server 2008.

expected value with respect to the probability of (fewer) scenarios in Ω_k. As investigated in [202], such a smaller subset of scenarios can be obtained by applying iteratively scenario reduction techniques to the set of vectors $\{b(\omega) - T(\omega)x\}_{\omega\in\Omega}$ (depending on the current decision vector). This is akin to approach (iii) above. Differently, scenario reduction/selection techniques in stochastic programming are in general employed in a static manner, considering the scenarios $\omega \in \Omega$ [52, 62, 159]. This yields the strategy (ii).

Lower oracles with on-demand accuracy is also possible for two-stage programs. Such oracles satisfy (1.2.5) with the additional assumptions in (1.2.6), where $0 \leq \eta_k \leq \eta$ and $f_k^{tar} \in \mathbb{R}$ are given parameters. The target f_k^{tar} can, for instance, be the (approximate) value of the function at a given previous iterate x_j; see [148] for more alternatives. An oracle with on-demand accuracy satisfying these assumptions for two-stage stochastic programs is described in Algorithm 1.

Algorithm 1 Oracle with on-demand accuracy for two-stage linear programs

▷ Initialization

1: Inputs: a point x_k, oracle error $\eta_k \geq 0$, target f_k^{tar} and
2: a set $\mathcal{D}$ with finitely many feasible points for (1.2.11)

▷ Fast approximation

3: **for** all $\omega \in \Omega$ **do**
4: Compute $u_{x_k,\omega} \in \arg\max \langle b(\omega) - T(\omega)x_k, u\rangle \quad \text{s.t.} \quad u \in \mathcal{D}$
5: **end for**

6: Set $f_{x_k} := \varphi(x_k) + \mathbb{E}[\langle b(\omega) - T(\omega), u_{x_k,\omega}\rangle]$ and $i := 0$

▷ Improving the approximation

7: **while** $f_{x_k} < f_k^{tar}$ and $i < |\Omega|$ **do**
8: Set $i = i + 1$ and pick up a (different) scenario $\omega \in \Omega$
9: Compute a η_k-solution $u_{x_k,\omega}$ to (1.2.11)
10: Add $u_{x_k,\omega}$ to $\mathcal{D}$ and consider removing an old point from $\mathcal{D}$
11: Set $f_{x_k} = \varphi(x_k) + \mathbb{E}[\langle b(\omega) - T(\omega), u_{x_k,\omega}\rangle]$
12: **end while**
Exit with f_{x_k} and $g_{x_k} := \nabla\varphi(x_k) - \mathbb{E}[T(\omega)^\top u_{x_k,\omega}]$

Notice that in the first part of the algorithm, denoted by fast approximation, there is no need to solve any linear programming problem. Since $\mathcal{D}$ contains only finitely many points, the procedure of piking up a point $u_{x_k,\omega}$ is essentially a matrix multiplication combined with a sorting process. When the oracle identifies that the approximate value f_{x_k} is greater than the given target f_k^{tar}, then it returns with the possibly poor information (f_{x_k}, g_{x_k}): there is no need to spend CPU time for computing exactly $f(x_k)$ once x_k is identified to be no better than a previous iterate yielding f_k^{tar}. Notwithstanding, the cheap and new oracle information (f_{x_k}, g_{x_k}) is possibly of poor quality, but can be employed to improve the cutting-plane model of f. Since the oracle is of the lower type (which is the case if $u_{x_k,\omega}$ is feasible for (1.2.11)), then the cutting-plane model never cuts off the function. Finally, we mention that between the lines 7-12 of Algorithm 1 the estimated value f_{x_k} is nondecreasing.

Specialized level bundle methods making use of an oracle as the one of Algorithm 1 has been applied to a large family of two-stage stochastic linear problems in [148]. We verified that the given methods could be up to 72% faster than the classical algorithm (L-Shaped) for that class of problems, without any accuracy loss. Later on it was showed by other colleagues in [209] that the level bundle method with on-demand accuracy reduces average solution time by 79% in an even larger battery of tests.

As another contribution to the two-stage setting, in the recent work [202] with Wim van Ackooij and Yongjia Song (Clemson University, USA), we proposed several strategies to solve two-stage stochastic linear programs by integrating the so-called adaptive partition-based approach with level bundle methods. A *partition-based formulation* is a relaxation of the original stochastic program, obtained by aggregating variables and constraints according to a scenario partition. As a theoretical contribution, we showed that for general two-stage stochastic linear programs with fixed recourse, there exists a particular partition whose corresponding partition-based master problem gives an optimal solution to the original stochastic program, and the size of this partition is independent of number of scenarios. In order to

try to determine such particular partition, partition refinements are guided by the optimal second-stage dual vectors $u_{x,\omega}$ computed at certain first-stage solutions x: for instance by applying clustering and scenario reduction techniques to the set of vectors $\{u_{x,\omega}\}_{\omega\in\Omega}$. This also can be seen an approach of type (iii) mentioned above.

Still in the two-stage stochastic setting, Chapter 4 presents the work [32] on a real-life two-stage stochastic energy problem from PETROBRAS, the Brazilian oil and gas company. The chapter models uncertainties in the long-term design and operation planning problem of the Brazilian natural gas network, and applies alternative (ii) above combined with bundle methods and two-stage decomposition to solve the underlying problem. The investigated strategies were implemented in the software `MONGE` that will assist, in 2019, the renegotiation of the 20-year gas supply contract between Brazil and Bolivia.

Multistage stochastic programs

Multistage stochastic programs explicitly model a series of decisions interplayed with partial observation of uncertainty. As in the two-stage setting, the typical approach to multistage stochastic programs is to approximate the underlying random process by using a scenario tree. This yields, in general, large-scale mathematical programming problems that can only be handled by specialized algorithms that employ decomposition techniques and very often sampling. Two popular decomposition schemes for handling multistage stochastic programs are the nested decomposition [27] and the stochastic dual dynamic programming (SDDP) [157]. By considering a dynamic formulation of the underling multistage linear problem, both strategies approximate the dynamic programming equations

$$Q_t(x_{t-1},\xi_t) := \left\{ \begin{array}{ll} \min\limits_{x_t\geq 0} & c_t^\top x_t + \mathcal{Q}_{t+1}(x_t) \\ \text{s.t.} & A_t x_t = b_t - B_t x_{t-1}\,, \end{array} \right. \tag{1.2.12}$$

where the stochastic process is denoted by $\xi_t(\omega) := (c_t(\omega), A_t(\omega), B_t(\omega), b_t(\omega))$ (for short $\xi_t := (c_t, A_t, B_t, b_t)$), $\mathcal{Q}_{t+1}(x_t) := \mathbb{E}[Q_{t+1}(x_t,\xi_{t+1})]$ for $t = T-1,\dots 1$, and $\mathcal{Q}_{T+1}(x_T) := 0$. The first-stage problem becomes

$$\min_{x_1\in X} f(x_1), \quad \text{with} \quad f(x_1) := c_1^\top x_1 + \mathcal{Q}_2(x_1)\,. \tag{1.2.13}$$

In general terms, this problem fits (1.1.1) with f a nonsmooth convex function and $\mathfrak{c}$ nonexistent. However, formulation (1.2.13) hides a *serious difficulty*: to date, there is no efficient and implementable oracle for the function f. Since f depends implicitly on a number of subproblems that in turn depend on other subproblems and so on, computing f (and a subgradient) is only possible via a recursive procedure that replaces the recourse functions $\mathcal{Q}_t$ by cutting-plane models $\check{\mathcal{Q}}_t$ ($\leq \mathcal{Q}_t$), $t = 2,\dots,T$:

$$\bar{x}_t \in \left\{ \begin{array}{ll} \arg\min\limits_{x_t\geq 0} & c_t^\top x_t + \check{\mathcal{Q}}_{t+1}(x_t) \\ \text{s.t.} & A_t x_t = b_t - B_t \bar{x}_{t-1}\,. \end{array} \right. \tag{1.2.14}$$

Such recursive procedure is split into two steps: a *forward* one defining feasible polices $\bar{x}_1,\dots,\bar{x}_T$ as above and estimating an upper bound for $f(\bar{x}_1)$, and *backward* step enriching the cutting models $\check{\mathcal{Q}}_t$, $t = T,\dots,2$, and computing a valid lower bound for the problem.

For numerical tractability, the stochastic process $\{\xi\}_{t=1}^T$ (with ξ_1 fixed) is represented by a scenario tree, as illustrated in the top of Figure 1.4. Each scenario $\xi^i = (\xi_1^i,\dots,\xi_T^i)$ consists of a path starting from the root node of the tree and going up to one of the leaf nodes. It is thus evident that the number of scenarios grows exponentially with the number of stages. Needless to say, the computational burden to solve (1.2.13) also grows with the number of scenarios. It is worth mentioning that the number of scenarios should be large enough to represent (by the Law of Large Numbers) well the stochastic process nature, but on the other hand the tree's size should be small to allow solving (1.2.13) in a reasonable CPU time. Important strategies to overcome (as much as possible) this difficulty are the *scenario reduction* techniques [159].

In a joint work with my PhD student Felipe Beltrán and my colleague Erlon Finardi, both from Federal University of Santa Catarina, Brazil, we investigated in [23] the scenario reduction algorithm of [109] applied to the Brazilian medium-term hydrothermal scheduling problem. Such a problem is modeled as a

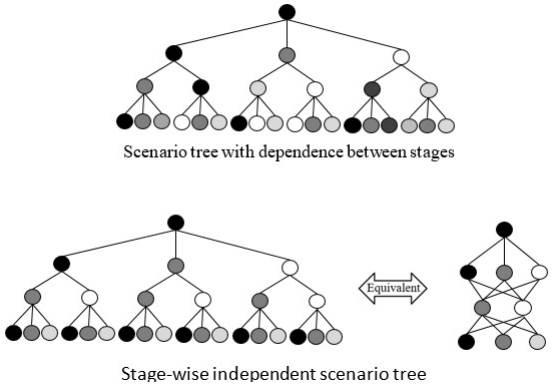


Figure 1.4: Stage-wise dependent and independent scenario trees. Both trees in the bottom of the figure represent the same 18 scenarios.

multistage stochastic program whose objective is to obtain an optimal operation policy over a planning horizon of few months by minimizing the expected cost of thermal generation. Water inflows to the hydro-plant reservoirs follow a stochastic process that is approximated by a multistage scenario tree. Under the assumption that the stochastic process is stage-wise independent, the scenario tree becomes a lattice, as illustrated in the bottom of Figure 1.4.

The considered scenario reduction algorithm employs the *quadratic distance process*, a particular instance of the *nested distance* of probability distributions proposed by George Ch. Pflug and Alois Pichler in [158]. Under the assumption that the scenario tree is stage-wise independent the algorithm of [109] can be simplified, permitting its usage to handle very large multistage scenario trees. For instance, a tree of seven stages and 4096 scenarios (denoted by original tree) for the Brazilian medium-term hydrothermal scheduling problem could be reduced to a smaller one (denoted by reduced tree) with 256 scenarios (93% of size reduction) in less than one second. With the original tree, the nested decomposition required almost three hours of processing, whereas the optimization problem issued by reduced tree was solved in less than one minute. When comparing the obtained solutions (decisions on power generation) in a out-of-sample simulation, the difference between the function values was less than 2%.

In general terms, numerical assessments of the investigated algorithm for stage-wise independent scenario trees applied to a smaller hydrothermal configuration extracted from the Brazilian system showed that reduced trees obtained by eliminating 80% of the scenarios provide approximate solutions to the problem with less than 1% of accuracy errors and CPU time reduction of around 90%. More details can be found in [23].

Another contribution to the field of multistage stochastic programming is the recent manuscript [199] that is the first work investigating level bundle methods for this class of problems. In that work we consider the nested decomposition and SDDP algorithm with a new scheme based on normal solutions for stabilizing iterates during the solution process. The given algorithms combine ideas from finite perturbation of convex programs and level bundle methods to regularize the so-called forward step of these decomposition methods. Essentially, instead of defining policies as in (1.2.14) in the forward step the manuscript proposes the following rule

$$\bar{x}_t \in \left\{ \begin{array}{ll} \arg\min\limits_{x_t} & \|x_t - x_t^{\mathtt{ref}}\|^2 \\ \text{s.t.} & x_t \in \left\{ \begin{array}{ll} \arg\min\limits_{y \geq 0} & \max\{c_t^\top y + \check{\mathcal{Q}}_{t+1}(y),\, f_t^{\text{lev}}\} \\ \text{s.t.} & A_t y = b_t - B_t \bar{x}_{t-1}\,, \end{array} \right. \end{array} \right. \tag{1.2.15}$$

where $x^{\mathtt{ref}}$ is a reference vector (e.g. $x^{\mathtt{ref}} = 0$) and f_t^{lev} is a given parameter estimating the optimal value $Q_t(\bar{x}_{t-1}, \xi_t)$ of (1.2.12). By employing the theory of finite perturbation of convex programs, the above subproblem becomes a quadratic programming problem (in contrast to (1.2.14) that is simply

linear). As a subproduct of this regularization scheme we showed that after finitely many steps the multistage regularized decomposition of [18] boils down to a particular case of SDDP, which seeks specific policies during the forward step. As a conclusion, the involving convergence analysis of the multistage regularized decomposition given in [18] can be alternatively replaced with a directly application of the well-understood SDDP analysis. Numerical experiments on a hydrothermal scheduling problem indicated that the new regularized algorithms are competitive with the state-of-the-art approaches in the area, outperforming the classical SDDP in many instances of the problem.

In summary, my main contributions to the field of stochastic programming with recourse consist in the algorithmic design and mathematical analysis of computational approaches to speed calculations for solving problems of this class. To this end, the combination of tools from convex analysis, bundle methods and scenario reduction/clustering proved crucial. Although very important, the class of stochastic programs with recourse is not the only type of stochastic problems that arise in practice: chance-constrained problems is of paramount importance when reliability is an issue.

1.2.2.2 Chance-constrained programming

In general terms, a stochastic optimization problem involving a chance constraint can be written in the form of (1.1.1), where $\mathfrak{c}$ depends on a probability function, e.g.

$$\mathfrak{c}(x) := p - \mathbb{P}[G(x,\omega) \geq 0]\,.$$

In this notation, $G : \mathbb{R}^n \times \Omega \to \mathbb{R}^m$ is a given mapping, $\omega : \Omega \to \mathbb{R}^m$ is a m-dimensional random vector defined on some probability space $(\Omega, \mathcal{A}, \mathbb{P})$, and $p \in (0,1]$ is a pre-specified probability level. The chance constraint

$$\mathfrak{c}(x) \leq 0 \quad \equiv \quad \mathbb{P}[G(x,\omega) \geq 0] \geq p \tag{1.2.16}$$

(also known as probabilistic constraint) expresses that the decision vector $x \in \mathbb{R}^n$ is feasible if and only if the random inequality system $G(x,\omega) \geq 0$ is satisfied with high enough probability [166]. Probability constraints are encountered in many engineering problems involving uncertain data. We can find applications in water management, telecommunications, electricity network expansion, hydro reservoir management, etc.

Since the function $\mathfrak{c}$ above is nonlinear, writing the constraint in the form $\mathfrak{c}(x) \leq 0$ makes (1.1.1) appear as a conventional nonlinear programming problem. However, this writing neglects a hidden difficulty: in most situations explicit values of $\mathfrak{c}$ are not available. Furthermore, often calculations are inexact, as computing the probability $\mathbb{P}[G(x,\omega) \geq 0]$ for a given point x typically involves multidimensional numerical integration and or (quasi) Monte Carlo methods. Other issues are that function $\mathfrak{c}$ sometimes fails to be convex and differentiable even when the mapping $-G$ is convex and differentiable on x [86, 87].

In general, chance-constrained models deal more explicitly with the probability distribution itself, whereas recourse models represent the randomness by finitely many scenarios and approximates the expectation by a finite weighted sum. For this reason, chance-constrained models can be mathematically more complicated than stochastic models with recourse, because they also explore more the information contained in the probability distribution. Naturally, there are also scenario-based strategies for chance-constrained programming.

A significant amount of effort has been made to understand the chance-constraint's properties such as (eventual) convexity [90] and (sub)differentiability [200], as well as to design specialized optimization methods to better explore and efficiently solve problems of this class. To this end, chance-constrained programming is in general split into different families of problems according to their randomness nature (discrete, continuous, elliptical probability distributions) and properties of the mapping G (linear, nonlinear, non-separable or separable, i.e., $G(x,\omega) = \mathfrak{g}(x) - \omega$, with $\mathfrak{g} : \mathbb{R}^n \to \mathbb{R}^m$).

By considering probabilistic constraints wherein the decision and random vector are separated, i.e. left/right-hand side uncertainty $\mathbb{P}[G(x,\omega) \geq 0 \equiv \mathbb{P}[\omega \leq \mathfrak{g}(x)]$, the Sklar's Theorem ensures that

the probability function can be represented by a composite function involving the mapping $\mathfrak{g}(x) = (\mathfrak{g}_1(x), \ldots, \mathfrak{g}_m(x))$, the marginal distributions F_i, $i = 1, ..., m$, of ω, and a Copula $\mathfrak{C} : [0,1]^m \to [0,1]$:

$$\mathbb{P}[\omega \leq \mathfrak{g}(x)] = \mathfrak{C}(F_1(\mathfrak{g}_1(x)), \ldots, F_m(\mathfrak{g}_m(x)))\,.$$

Copula is a multivariate probability distribution for which the marginal-probability distribution of each variable is uniform.

In the joint paper [11] with Wim van Ackooij we modeled probabilistic constraints with Copulæ and investigated eventual convexity of the feasible set

$$X(p) := \{x \in X : \mathfrak{C}(F_1(\mathfrak{g}_1(x)), \ldots, F_m(\mathfrak{g}_m(x))) \geq p\}\,.$$

Eventual convexity means that although the set $X(p)$ is not convex for all $p \in (0,1]$, there exists a threshold $p^* \in (0,1]$ depending on $\mathfrak{C}$, $\mathfrak{g}$ (possessing generalized concavity properties) and marginals F_i $i = 1, \ldots, m$, such that $X(p)$ is a convex set for all $p \geq p^*$. This is a property of practical interest, since one generally cares about convexity of $X(p)$ for p nearly 1: a stochastic system of inequalities should be satisfied with a high probability. In [11] we showed that the broad class of Archimedean Copulæ yields eventual convexity of the set $X(p)$ (provided certain level sets of $\mathfrak{g}_i$ are convex). Naturally, eventual convexity of $X(p)$ does not imply that the function $x \mapsto -\mathfrak{C}(F_1(\mathfrak{g}_1(x)), \ldots, F_m(\mathfrak{g}_m(x)))$ is convex. In order to solve the resulting problem (1.1.1) with constraint

$$\mathfrak{c}(x) \leq 0 \quad \equiv \quad p - \mathfrak{C}(F_1(\mathfrak{g}_1(x)), \ldots, F_m(\mathfrak{g}_m(x))) \leq 0$$

the work [11] proposed the first level bundle method able to deal with a constraint function possessing generalized convexity properties (e.g. quasi-convexity). More details on this subject are given in Chapter 5.

Still in the setting of separable probability function, if the random variable is uni-dimensional (i.e., $m = 1$), then the problem of minimizing a function f over $X \cap \{x \in \mathbb{R}^n : \mathbb{P}[\omega \leq \mathfrak{g}(x)] \geq p\}$ can be stated in a *deterministic form* by employing the concept of p-quantile of the probability distribution $F(\mathfrak{g}(x)) = \mathbb{P}[\omega \leq \mathfrak{g}(x)]$:

$$\min_{x \in X} \; f(x) \quad \text{s.t.} \quad \mathfrak{g}(x) \geq F^{-1}(p)\,.$$

Computing the p-quantile $F^{-1}(p)$ is not a difficult task (when $m = 1$) and can be accomplished by standard computational toolboxes of probability and statistics for a wide range of uni-variate distributions. Once $F^{-1}(p)$ is computed, the above becomes an ordinary deterministic optimization problem that can be solved by standard algorithms.

In the multidimensional setting $m > 1$, the concept of p-quantile was extended to what is called p-efficient point:
A vector $v \in \mathbb{R}^m$ is called a p-efficient point of the probability distribution of ω, if $\mathbb{P}[\omega \leq v] \geq p$ and there is no $y \leq v$, $y \neq v$ such that $\mathbb{P}[\omega \leq y] \geq p$.
Instead of a single value $F^{-1}(p)$, the set $\mathcal{V}$ of p-efficient points can contain infinitely many of them (only a finite number if the support of the probability distribution is finite). With this definition, problem (1.1.1) with $\mathfrak{c}$ given in (1.2.16) can be rewritten as

$$\min_{x,v} \; f(x) \quad \text{s.t.} \quad \mathfrak{g}(x) \geq v\,,\; x \in X \;\text{ and }\; v \in \mathcal{V}.$$

From the algorithmic point of view, the main advantage of this formulation is that it exempts the need of computing subgradients of the probability function. But this interesting feature comes with a price: to compute a p-efficient point one needs to solve a combinatorial subproblem whose dimension depends on the size of the sample generated to approximate the uncertain parameters. What if, to speed up calculations, we consider approximated p-efficient points? What type of result regarding solution quality can we expect?

To answer these questions, in [4] we iteratively computed points from $\mathcal{V}$ by inexactly solving the combinatorial subproblems and, by adopting a dual point of view, we developed a solution framework that

includes and extends various existing formulations. Our approach, which can be applied to both discrete and continuous random variables, represents a contribution in three fronts. First, by extending the theory on inexact bundle methods developed in [149] to the primal-dual setting, we revealed the impact of inexactness in primal terms. Second, we designed new on-demand accuracy approaches which performed the best in our numerical experiments. Third, thanks to the unifying view, we showed convergence for a generalization of both the Regularized Dual Decomposition and the Progressive Augmented Lagrangian algorithm [56, 59]. This work is discussed in details in Chapter 6.

If probability constraints is not separable, then the p-efficient point strategy is not suitable and other solving strategies must come into play. In the joint work with Wim van Ackooij and Antonio Frangioni (University of Pisa) [203] we investigated the celebrated (generalized) Benders' decomposition approach applied to probability constraints (not necessarily separable) with finite support Ω. Binary variables were employed to handle the finite set of scenarios Ω:

$$\begin{cases} \min\limits_{x\in X, z} & f(x) \\ \text{s.t.} & G(x,\omega) \leq z_\omega\, M \\ & \langle \pi, z\rangle \leq 1-p \\ & z_\omega \in \{0,\,1\} \;\; \forall \omega \in \Omega\,, \end{cases}$$

where $M \in \mathbb{R}^m_+$ is a vector composed of large enough constants, and π is the vector of probabilities. We then split these (binary) variables into a first stage of decisions, yielding a value function resulting from the optimization of the easier (but not trivial) subproblem:

$$v(z) := \min_{x\in X}\; f(x) \quad \text{s.t.} \quad G(x,\omega) \leq z_\omega\, M \;\; \forall \omega \in \Omega\,.$$

By defining this subproblem, the master program then becomes purely combinatorial

$$\begin{cases} \min\limits_{z} & v(x) \\ \text{s.t.} & \langle \pi, z\rangle \leq 1-p \\ & z_\omega \in \{0,\,1\} \;\; \forall \omega \in \Omega\,. \end{cases}$$

The approach proposed in [203] combines stabilization in such a Benders's decomposition in two ways: via a trust region in the ℓ_1 norm, or via a level constraint and inexact function computation (solution of the subproblems). Managing both features simultaneously required a non trivial convergence analysis. We provided it under very weak assumptions on the handling of the parameters controlling the informative on-demand inexact oracle corresponding to the subproblem, strengthening earlier known results [9, 148].

Final comments on this chapter.
By starting with the presentation of my fields of action in mathematical programming, a general discussion on my scientific contributions has been presented. As we could see, the core of my research lies in the algorithmic aspects of nonsmooth optimization (convex, mixed-integer, DC) and stochastic programming (with recourse and with probabilistic constraint). Energy management problems as the ones of [9, 23, 32, 49] have motivated most of the research topics discussed above. In what follows, the five articles [4], [11], [32], [50] and [51] on stochastic programming and nonsmooth optimization are presented in details, except for the mathematical proofs that can be found in the original publications.

Chapter 2

Target radius method for nonsmooth convex optimization

This chapter is extract from the following paper, where the convergence analysis of the *Target Radius Method* (TRM) of [153] is presented:

> W. de Oliveira
> *Target radius methods for nonsmooth convex optimization.*
> **Operations Research Letters**, 2017, volume 45, issue 6, pp. 659-664.

The work shows that the method belongs to the level bundle family, and significantly improves the TRM of [153] by handling particular classes of nonlinearly constrained convex problems and bilevel programs. The paper [50] also provides an original *limited-memory* strategy that can employed in bundle methods. Moreover, it is shown that the given TRM possesses dimension-independent iteration complexity and optimal (in the large-scale case) rate-of-convergence results for the minimization of a convex function over a Euclidean ball, a standard simplex, and other domains.

2.1 Introduction

We study the *Target Radius Method* – TRM – proposed in [153] for minimizing a nonsmooth convex function $f : \mathbb{R}^n \to \mathbb{R}$ over a convex set $X \neq \emptyset$:

$$\min_{x \in X} f(x). \tag{2.1.1}$$

The method combines the central cutting-plane algorithm of [64] with level bundle methods [30, 106, 123, 148], and defines every trial point (candidate solution) as the center of the closest sphere (with a given radius) inscribed in a certain polyhedron issued by a set of linearizations of function f.

Adam Ouorou briefly proposed TRM as a variant of the *Proximal Chebychev Center Cutting Plane Algorithm* (pc^3pa) in [153, page 260]. No convergence analysis was given. In this work, we provide a convergence analysis with worst-case bound for complexity results and propose a handy rule to update the method's most important parameter: the inscribed sphere's radius. Furthermore, we extend TRM to handle two additional classes of convex nonsmooth optimization problems described below.

2.1.1 Problems of interest and main assumptions

Throughout this paper we make the assumption that $\mathfrak{c} : \mathbb{R}^n \to \mathbb{R}$ is a continuous differentiable and strongly convex function on X with parameter $\rho > 0$, w.r.t. the norm $\|\cdot\|_p$ ($p \in [1, \infty]$), that is

$$\mathfrak{c}(x) \geq \mathfrak{c}(y) + \langle \nabla \mathfrak{c}(y), x - y \rangle + \frac{\rho}{2} \|x - y\|_p^2 \quad \forall\, x, y \in X\,. \tag{2.1.2}$$

In addition to (2.1.1), we deal with problems of the form

$$\min_{x \in X} \; f(x) \quad \text{s.t.} \quad \mathfrak{c}(x) \leq \delta\,. \tag{2.1.3}$$

Nonlinearly-constrained optimization problems as (2.1.3) arise, for instance, as subproblems in *Trust-Region Methods* whose nonlinear constraint $\mathfrak{c}$ is a Bregman function (e.g. $\|x - \hat{x}\|_2^2$ and $\hat{x} \in \mathbb{R}^n$ is a given stability center). Notice that setting (2.1.1) can be recovered from (2.1.3) by setting $\delta = +\infty$, making the constraint $\mathfrak{c}(x) \leq \delta$ in (2.1.3) superfluous. With this in mind, we provide a unified presentation handling problems (2.1.1) and (2.1.3) in an identical manner, yielding the same complexity result for both classes of problems.

After some simple modifications, TRM is extended to handle the following particular class of bilevel problems

$$\min_{x \in \mathbb{R}^n} \; \mathfrak{c}(x) \quad \text{s.t.} \quad x \in \arg\min_{y \in X} f(y)\,. \tag{2.1.4}$$

For the particular case $\mathfrak{c}(x) = \|x\|_2^2$, the above problem is the so-called *minimal norm solution problem*, see [5]. A more general (yet particular) bilevel program is investigated in [187] under the light shed by bundle methods and penalty functions. Differently from [187], we do not employ any penalization approach.

In summary, we propose two algorithms: one handling problems (2.1.1) and (2.1.3), and another one dealing with problem (2.1.4). Both algorithms generate a subsequence of candidate solutions converging to an optimal solution of the underlying optimization problem. Trial points are obtained by solving a *master problem* of the form

$$\min_{x \in X} \; \mathfrak{c}(x) \quad \text{s.t.} \quad Gx + \alpha \leq 0, \tag{2.1.5a}$$

with G and α given matrix and vector representing a cutting-plane approximation of f. We focus on a particular class of problems whose constraint function $\mathfrak{c}$ and feasible set X are simple enough so that solving either problem (2.1.5a) or its dualized version (yielding the dual function of (2.1.5a))

$$\min_{x \in X} \; \mathfrak{c}(x) + \langle \lambda, Gx \rangle \tag{2.1.5b}$$

is a simple (or at least computationally cheap) task. If the considered problem is of type (2.1.1), then we have freedom to choose an appropriated function $\mathfrak{c}$ (combined with X) to make the master problem (2.1.5a) (or (2.1.5b)) easy to solve; and/or to obtain nearly dimension-independent (and nearly optimal in the large-scale case) rate-of-convergence results; see [25, 110].

2.2 Target radius method

As our first algorithm can handle both problems (2.1.1) and (2.1.3) in a similar manner, we will focus on the more general problem (2.1.3), keeping in mind that problem (2.1.1) can be recovered by setting $\delta = \infty$. As a result, we will not make any distinction between these settings (except when needful).

The proposed method for solving problem (2.1.3) generates a sequence of iterates belonging to

$$\{x \in X : \mathfrak{c}(x) \leq \delta\}.$$

For every point x_k an oracle is called to compute $f(x_k)$ and a subgradient $g_k \in \partial f(x_k)$. With such information, the method creates the linearization

$$f(x_k) + \langle g_k, x - x_k \rangle \leq f(x)\,,$$

where the inequality follows from convexity of f. At iteration k, let $f_k^{\text{up}} := f(x^{\text{best}}) := \min_{j \leq k} f(x_j)$. Since trial points are feasible, f_k^{up} is a valid upper bound for problem (2.1.3). Suppose a valid lower bound f_k^{low} is known; we compute the optimality gap and level parameter as

$$\Delta_k := f_k^{\text{up}} - f_k^{\text{low}} \quad \text{and} \quad f_k^{\text{lev}} := f_k^{\text{low}} + \gamma_1 \Delta_k, \ \text{with } \gamma_1 \in (0,1)\,.$$

Note that $\Delta_k \geq 0$ and $f_k^{\text{lev}} \in (f_k^{\text{low}}, f_k^{\text{up}})$. The algorithms below stop when Δ_k is smaller than a given tolerance $\text{Tol} > 0$.

2.2.1 The Ouorou's algorithm

Given a radius $\sigma_k \geq 0$ at iteration k, the algorithm proposed in [153] defines the next trial point x_{k+1} as a solution of

$$\left\{ \begin{array}{ll} \min\limits_{x \in X} & \frac{1}{2} \left\| x - x^{\text{best}} \right\|_2^2 \\ \text{s.t.} & f(x_j) + \langle g_j, x - x_j \rangle + \sigma_k \lambda_j \quad \leq f_k^{\text{lev}}, \ \ \forall\, j = 1, \ldots, k\,, \end{array} \right. \tag{2.2.1}$$

with $\lambda_j := 1 + \sqrt{1 + \|g_j\|_2^2}$, that is, the algorithm searches for the sphere of radius σ_k inscribed in

$$L_k := \left\{ (x,r) \in \mathbb{R}^{n+1} \,\middle|\, \begin{array}{l} f(x_j) + \langle g_j, x - x_j \rangle \leq r, \ \ \forall\, j \leq k \\ r \leq f_k^{\text{lev}} \end{array} \right\},$$

whose x-component of the center (x_{k+1}, r_{k+1}) is closest to the point x^{best}; see [153] for more details. Notice that (2.2.1) is a particular case of the master problem (2.1.5a) with $\mathfrak{c}(x) = \frac{1}{2}\|x - \hat{x}\|_2^2$ and constraints properly written. If $\sigma_k \equiv 0$ for all iterations k, then subproblem (2.2.1) becomes the master problem of level bundle methods [30, 106, 123, 148].

The updating rule for σ_k proposed in [153] is: $\sigma_{k+1} = \gamma \sigma_k^{\max}$, $\gamma \in (0,1)$, where $\sigma_k^{\max}$ is the radius of the Chebychev sphere in L_k, which then needs solving an extra linear subproblem.

2.2.2 The new algorithm

In [25] the authors show that better complexity results can be obtained by replacing the quadratic objective function in level methods' master problems with appropriate Bregman-like distances fitting the structure of X. In this work we go further and replace the objective function in (2.2.1) with a (differentiable) strongly convex function $\mathfrak{c}$ on X. To this end, we consider the following set

$$V := \left\{ x \in X : \langle \nabla \mathfrak{c}(x), x - z \rangle \leq 0 \quad \forall\, z \in S^* \right\}, \tag{2.2.2}$$

where $S^* \neq \emptyset$ is the solution set of (2.1.3). We associate to $\|\cdot\|_p$ ($p \in [1, \infty]$) its dual norm $\|\cdot\|_q$ with $q = p/(p-1)$, and the convention that $1/0 = \infty$ and $\infty/\infty = 1$. Given $\sigma_k \geq 0$, the next trial point x_{k+1} solves

$$\left\{ \begin{array}{ll} \min\limits_{x \in X} & \mathfrak{c}(x) \\ \text{s.t.} & \langle \nabla \mathfrak{c}(\tilde{x}_k), \tilde{x}_k - x \rangle \leq 0 \\ & f(x_j) + \langle g_j, x - x_j \rangle \leq f_k^{\text{lev}} - \sigma_k [1 + \|(1, g_j)\|_q] \\ & \quad j \in J_k \end{array} \right. \tag{2.2.3}$$

where $\tilde{x}_k$ is either x_k or $x_0 \in V$, and $J_k \subset \{0, \ldots, k\}$. The choice of $\tilde{x}_k$ is automatically performed by our algorithms in order to obtain a limited-memory method (i.e., when J_k is required to be uniformly bounded). Note that if the given norm $\|\cdot\|_p$ is the Euclidean one (i.e., $p = 2$), then $\|(1, g_j)\|_2 =$

$\sqrt{1+\|g_j\|_2^2}$ as in [153]. Our algorithms, however, do not restrict themselves to the Euclidean norm. Alternatively, we may replace $1+\|(1,g_j)\|_q$ in (2.2.3) with $\|g_j\|_q$: in this case, the sphere of radius σ_k will be inscribed in

$$\{x \in \mathbb{R}^n : f(x_j) + \langle g_j, x - x_j\rangle \leq f_k^{\text{lev}}\}$$

rather than in L_k.

Let $\Lambda_k := \max_{j=0,\ldots,k}\{1+\|(1,g_j)\|_q\}$. The following two lemmas ensure that if (2.2.3) is infeasible, then $f_k^{\text{lev}} - \sigma_k\Lambda_k$ is a lower bound for the optimal value of (2.1.3).

Lemma 2.2.1. *([50, Lemma 1]). Let f^* and $S^* \neq \emptyset$ be the optimal value and solution set of problem (2.1.3), $\mathbb{X}_k$ be the feasible set of (2.2.3), and V as in (2.2.2). If $f_k^{\text{lev}} - \sigma_k\Lambda_k \geq f^*$ and $\tilde{x}_k \in V$, then $S^* \subset \mathbb{X}_k$ and the next iterate x_{k+1} solution of (2.2.3) is well-defined, belongs to V and satisfies $\mathfrak{c}(x_{k+1}) \leq \delta$.*

The following lemma gives a handy rule to update lower bounds of f^*.

Lemma 2.2.2. *([50, Lemma 2]). Given $x_{k_0} \in V$ and $k_0 \geq 0$, let the sequence $\{x_i\}$ be generated by solving (2.2.3), and let $k > k_0$. Suppose that $\{f_i^{\text{lev}} - \sigma_i\Lambda_i\}_{i=k_0}^k$ is a nonincreasing sequence and $\tilde{x}_i = x_i$ for all $i = k_0, \ldots, k$. If either $\mathbb{X}_k = \emptyset$ or $\mathfrak{c}(x_{k+1}) > \delta$, then $f^* > f_i^{\text{lev}} - \sigma_i\Lambda_i$ for some $i \in \{k_0, 1, \ldots, k\}$ and, in particular, $f_k^{\text{lev}} - \sigma_k\Lambda_k$ is a lower bound for f^*.*

As a result of Lemma 2.2.2, we update the lower bound f_k^{low} by the following rule: set $f_{k+1}^{\text{low}} = f_k^{\text{lev}} - \sigma_k\Lambda_k$ whenever $\mathbb{X}_k = \emptyset$ or $\mathfrak{c}(x_{k+1}) > \delta$. In order to ensure the above rule provides valid lower bounds along the iterative process we need to start with $x_0 \in V$ and reset $\tilde{x}_k$ to a point in V whenever f_k^{low} is updated. This is automatically done by Algorithm 2.

Algorithm 2 Target Radius Algorithm

Step 1. Let $x_0 = \arg\min_{x\in X} \mathfrak{c}(x)$, compute $f(x_0)$, $g_0 \in \partial f(x_0)$. Define $f_0^{\text{up}} = f(x_0)$, $x^{\text{best}} = x_0$, $\Lambda_0 = 1+\|(1,g_0)\|_q$ and $J_0 = \{0\}$. Determine $f_0^{\text{low}} < f^*$, choose $\gamma_1 \in (0,1)$, $\gamma_2 \in [0,1)$, Tol ≥ 0, $m \geq 2$ and set $k = l = k(0) = 0$.

Step 2. Define $\Delta_k = f_k^{\text{up}} - f_k^{\text{low}}$.
If $\Delta_k \leq$ Tol, stop and return x^{best} and $f(x^{\text{best}})$.

Step 3. Set $f_k^{\text{lev}} = f_k^{\text{low}} + \gamma_1\,\Delta_k$ and $\sigma_k = \gamma_2\gamma_1\Delta_k/\Lambda_k$.
Set $\tilde{x}_k = x_k$ if $k > k(l)$, or $\tilde{x}_k = x_0$, if $k = k(l)$.

Step 4. Try to solve (2.2.3) to compute x_{k+1}.
If (2.2.3) is infeasible or $\mathfrak{c}(x_{k+1}) > \delta$, then set $f_{k+1}^{\text{low}} = f_k^{\text{lev}} - \sigma_k\Lambda_k$, $k(l+1) = k+1$, $l = l+1$, $f_{k+1}^{\text{up}} = f_k^{\text{up}}$, $\Lambda_{k+1} = \Lambda_k$ and go to Step 6. Otherwise set $f_{k+1}^{\text{low}} = f_k^{\text{low}}$ and continue.

Step 5. Compute $f(x_{k+1})$, $g_{k+1} \in \partial f(x_{k+1})$ and set $\Lambda_{k+1} = \max\{1+\|(1,g_{k+1})\|_q, \Lambda_k\}$.
If $f(x_{k+1}) < f_k^{\text{up}}$, set $f_{k+1}^{\text{up}} = f(x_{k+1})$ and $x^{\text{best}} = x_{k+1}$. Otherwise $f_{k+1}^{\text{up}} = f_k^{\text{up}}$.

Step 6. Choose $\tilde{J} \subset J_k$ such that $|\tilde{J}| \leq m-2$. Set $J_{k+1} = \tilde{J} \cup \{0, k+1\}$, $k = k+1$ and go back to Step 2.

Remark 2.2.3. *The initial point x_0 solution of the strongly convex program $\min_{x\in X} \mathfrak{c}(x)$ is readily available in some important applications. This is the case of trust-region subproblems and signal/image processing where $\mathfrak{c} : X \to \mathbb{R}_+$ is a Bregman function $\mathfrak{c}(x) = \mathfrak{s}(x) - \mathfrak{s}(\hat{x}) - \langle\nabla\mathfrak{s}(\hat{x}), x - \hat{x}\rangle$ defined with $\hat{x} \in X$. The distance-generating-function $\mathfrak{s} : X \to \mathbb{R}$ is assumed to be a continuous differentiable and strongly convex on X with parameter $\rho > 0$, w.r.t. the norm $\|\cdot\|_p$. In such setting $x_0 = \hat{x}$ is the unique minimizer of $\mathfrak{c}$ over X. However, if x_0 is not known in advance, we may find it by employing an out-of-shelf algorithm for smooth optimization to the problem $\min_{x\in X} \mathfrak{c}(x)$. The first-order optimality condition yields $\langle\nabla\mathfrak{c}(x_0), x - x_0\rangle \geq 0$ for all $x \in X$. In particular, $x_0 \in V$ and $\mathfrak{c}(x_0) \leq \delta$ because (2.1.3) is feasible.*

Suppose an initial lower bound f_0^{low} is not available. If X is a compact set we may compute f_0^{low} by solving $\min_{x\in X}\langle g_0, x\rangle$, justified by the inequality $f(x_0) + \langle g_0, x - x_0\rangle \leq f(x)$.

Notice that the choice $\gamma_2 = 0$ is possible. In this case Algorithm 2 becomes a level bundle algorithm, [30]. Choosing $\gamma_2 > 0$ yields deep cuts, cutting off a larger region of the feasible set. According to Step 3 of the algorithm, the vector $\tilde{x}_k$ is reset to x_0 whenever the lower bound is updated. Otherwise $\tilde{x}_k$ is the current point x_k. This strategy is crucial to ensure that $f_k^{\text{low}} \leq f^*$ for all k (c.f. Lemma 2.2.2).

If function $\mathfrak{c}$ and feasible set X have favorable structures, the natural way is to solve (2.2.3) directly by some specific (QP, conic etc.) solver. If it is not the case, then we may solve the dual problem of (2.2.3):

$$\max \theta(\lambda) \quad \text{s.t.} \quad \lambda \in \mathbb{R}_+^{|J_k|+1}, \quad \text{with } \theta(\lambda) := \langle \lambda, \alpha \rangle + \min_{x \in X} \mathfrak{c}(x) + \sum_{j \in J_k} \lambda_j \langle g_j, x \rangle - \lambda_0 \langle \nabla \mathfrak{c}(\tilde{x}_k), x \rangle. \quad (2.2.4)$$

We assume the inner subproblem defining $\theta(\lambda)$ is easy to solve, having possibly an explicit expression for computing its solution. The task of maximizing θ can be performed by a bundle algorithm [30, 148, 149]. Suppose $\delta < \infty$. If for a given λ one has $\theta(\lambda) > \delta$, then the optimal value of (2.2.4) is strictly greater than δ. By the weak duality the optimal value of (2.2.3) (possibly $+\infty$ when the master problem is infeasible) is also strictly greater than δ, and thus x_{k+1} is not feasible (by Lemma 2.2.2) for (2.1.3). We shall thus stop the optimization process of (2.2.4) and update the lower bound f_k^{low} according to Step 4.

Step 5 ensures that size of J_k is overall bounded by $m \geq 2$, a constant that is chosen at Step 1. A moderate value of m makes problem (2.2.3) (and (2.2.4)) easier to solve, however a larger value for m provides a better approximation of f and possibly decreases the number of iterations to solve problem (2.1.3). The bundle management of Step 6 always keep the oracle information of x_0. This is not crucial for the algorithm convergence, but it is useful to prove its low complexity (see Proposition 2.3.3 below).

2.3 Convergence analysis

Throughout this section we denote $\kappa := 1 - \gamma_1(1 - \gamma_2) \in (0, 1)$ and $\lambda_j := 1 + \|(1, g_j)\|_q$. We recall that f is a finite-valued convex function, and hence its subdifferential $\partial f(x)$ is nonempty, convex, and bounded for all x. Let

$$Y := X \cap \{x \in \mathbb{R}^n : \mathfrak{c}(x) \leq \delta\} \quad \text{be a compact set.} \quad (2.3.1)$$

Then there exists a constant $\Lambda > 0$ such that $\|g\|_q \leq \Lambda - 1$ for all $g \in \partial f(x)$ and all $x \in Y$. As a result, the value $\Lambda_k = \max_{j=0\ldots,k} \lambda_j$ in Algorithm 2 is an approximation of Λ, which we assume w.l.o.g. to satisfy $\Lambda_k \leq \Lambda$ for all k. We will also make use of the following useful index set

$$K_l := \{k(l), k(l) + 1, \ldots, k(l+1) - 1\}, \ \forall\, l = 0, 1, \ldots, \quad (2.3.2)$$

splitting the iterative process of Algorithm 2 into cycles.

Lemma 2.3.1. *([50, Lemma 3]). Step 4 of Algorithm 2 ensures that $\Delta_{k(l+1)} \leq \kappa \Delta_j$, for all $j \in K_l$ and all $l = 0, 1, \ldots$ Moreover, $\{f_j^{\text{lev}} - \sigma_j \Lambda_j\}_j$ is nonincreasing for all $j \in K_l$.*

The convergence analysis given below, inspired in [106], consists in showing that every cycle K_l in (2.3.2) is finite.

Lemma 2.3.2. *([50, Lemma 4]). Let $k \in K_l$ and assume that x_{k+1} is well defined. Then $\|x_{k+1} - x_k\|_p \geq \kappa \Delta_k / \Lambda$ if $k > k(l)$, and $\|x_{k+1} - x_0\|_p \geq \kappa \Delta_k / \Lambda$ if $k = k(l)$.*

Proposition 2.3.3. *([50, Proposition 5]). Assume (2.3.1) and let $\mathfrak{D}$ be $\delta - \mathfrak{c}(x_0) \geq 0$ if $\delta < \infty$, or $\max_{x \in X} \mathfrak{c}(x) - \mathfrak{c}(x_0) \geq 0$ otherwise. If $k \in K_l$ and $\Delta_k > 0$, then*

$$k - k(l) \leq \frac{2\mathfrak{D}}{\rho} [\Lambda / (\kappa \Delta_k)]^2.$$

Let k be the largest index in K_l. Then, cardinality $|K_l|$ of K_l is $k - k(l) + 1$, which is bounded by $\frac{2\mathfrak{D}}{\rho}[\Lambda/(\kappa\Delta_k)]^2 + 1$ (c.f. Proposition 2.3.3). However, as the oracle is not called at iterations $k(l)$, for $l = 1, 2, \ldots$, (see Step 4 of the algorithm) the number of oracle calls in every index set K_l (except K_0) is $|K_l| - 1$. In what follows we present a bound on the maximum number of oracle calls required by Algorithm 2 to obtain a Tol-solution.

Theorem 2.3.4. *([50, Theorem 6]). Consider Algorithm 2, assume* (2.3.1) *and let* $\mathfrak{D}$ *be as in Proposition 2.3.3. Then, to reach an optimality gap smaller than* $\mathrm{Tol} > 0$ *it is enough to perform at most*

$$\frac{2\mathfrak{D}}{\rho}\left(\frac{\Lambda}{\mathrm{Tol}}\right)^2 \frac{1}{\kappa^2(1-\kappa^2)} + 1 \text{ oracle calls.}$$

Remark 2.3.5. *Let us now consider the case in which* $\mathfrak{D}$ *in Proposition 2.3.3 is zero. In this case, Theorem 2.3.4 states that Algorithm 2 will perform only one oracle call (at the point* x_0*). This is as expected, as discussed below:*

- *suppose* $\delta < \infty$*, i.e., we are in the setting of problem* (2.1.3)*. Notice that* $\mathfrak{D} = 0$ *implies* $\mathfrak{c}(x_0) = \delta$*. Since* $\mathfrak{c}$ *is a strongly convex function and* x_0 *is its minimizer over* X*, we conclude that the feasible set of problem* (2.1.3) *is the singleton* $\{x_0\}$*;*
- *suppose* $\delta = \infty$*, i.e., problem* (2.1.1) *is considered. In this case,* $\mathfrak{D} = 0$ *implies* $\max_{x\in X} \mathfrak{c}(x) = \mathfrak{c}(x_0) = \min_{x\in X} \mathfrak{c}(x)$*, and therefore the feasible set of problem* (2.1.1) *is the singleton* $\{x_0\}$*.*

In both cases Algorithm 2 will not perform more than one oracle call. However, if $\Delta_0 > 0$ *it will require at most* $\frac{\log(\mathrm{Tol}/\Delta_0)}{\log(\kappa)}$ *iterations until proving that* x_0 *is a* Tol*-solution.*

2.3.1 Special setups and asymptotic result

The best possible bound in Theorem 2.3.4 is $\frac{8\mathfrak{D}}{\rho}\left(\frac{\Lambda}{\mathrm{Tol}}\right)^2 + 1$, achieved when $\kappa = [1 - \gamma_1(1-\gamma_2)] = \sqrt{2}/2$. When problem (2.1.1) is considered, we have the possibility to choose a suitable function $\mathfrak{c}$ to adjust to the geometry of the feasible set X, minimizing thus the fraction $\frac{\mathfrak{D}}{\rho}$. This allows us to get nearly dimension-independent (and nearly optimal in the large-scale case) rate-of-convergence results for minimization of a convex function over a Euclidean ball, a standard simplex, and other domains [25, § 2.3].

2.3.1.0.1 Mahalanobis distance Given a squared, symmetric and positive definite matrix Q, suppose function $\mathfrak{c}$ is given by $\mathfrak{c}(x) = \frac{1}{2}\langle x - \hat{x}, Q(x - \hat{x})\rangle$, with $\hat{x} \in X$ a given stability center. If X is a polyhedron, then (2.2.3) is a QP. If X is the whole space $\mathbb{R}^n$, then the dual problem (2.2.4) is a QP with $|J_k| + 1$ variables, being easy to solve if $|J_k|$ is of moderate size (say a few hundreds). If $Q = I$ and $\delta = 1$, then $\rho = 1$, $x_0 = \hat{x}$ and $\mathfrak{D} = \delta - \mathfrak{c}(x_0) = \delta = 1$. In this case, the bound on the number of oracle calls is dimension independent: $8\left(\frac{\Lambda}{\mathrm{Tol}}\right)^2 + 1$.

2.3.1.0.2 Generalized Kullback-Leibler (GKL) divergence Consider problem (2.1.1) with $X = \{x \in \mathbb{R}^n_+ : \sum_{i=1}^n x_i = 1\}$. Then the function of choice is the GKL divergence $\mathfrak{c}(x) = \mathfrak{s}(x) - \mathfrak{s}(\hat{x}) - \langle \nabla\mathfrak{s}(\hat{x}), x - \hat{x}\rangle$, with $\hat{x}$ a fixed point in X and $\mathfrak{s}(x) = \sum_{i=1}^n (x_i + \nu n^{-1}) \log(x_i + \nu n^{-1})$, for a given positive tolerance $\nu \approx 0$. In this setup, computing the solution of (2.2.3) amounts at finding a root of a certain unidimensional equation (e.g., [25, page 421], which is an easy task. Moreover, it can be shown that $\rho = (1+\nu)^{-1}$, $\|\cdot\|_p = \|\cdot\|_1$ and $\mathfrak{D} \leq (1+\nu)[1+\ln(n(1+\nu)/\nu)]$, [25, § 2.3]. The initial point in Algorithm 2 is $x_0 = \hat{x}$.

In contrast to [25], Algorithm 2 (a) considers "deep cuts" yielded by radius σ_k in (2.2.3); (b) does not require solving an extra (linear or nonlinear) subproblem to update lower bounds; (c) has the possibility to consider more general functions $\mathfrak{c}$ (not only Bregman functions as in [25]). Moreover, Algorithm 2 is able to solve the bilevel problem (2.1.4) under the more restrictive assumption (2.3.3) below.

Theorem 2.3.6. *([50, Theorem 7]). Assume* (2.3.1) *and let* $\mathrm{Tol} = 0$ *in Algorithm 2. Then* $\lim \Delta_k = 0$ *and any cluster point of the sequence* $\{x_k^{\mathrm{best}}\}$ *converges to a solution of* (2.1.3)*.*

Furthermore, suppose that $\delta = \infty$ *and*

$$f_k^{\text{lev}} - \sigma_k \Lambda_k \geq f^* \quad \forall\, k = 0, 1, 2, \ldots \tag{2.3.3}$$

Then $\{x_k^{\text{best}}\}$ *converges to the unique solution of* (2.1.4).

Assumption (2.3.3) is satisfied, for instance, when the optimal value f^* of (2.1.1) is known and we take $f_0^{\text{low}} = f^*$ (and therefore $f_k^{\text{low}} = f^*$ for all k). This situation is, nevertheless, unrealistic in many problems of practical interest. Below we propose a variant of Algorithm 2 that is able to solve (2.1.4) without requiring assumption (2.3.3).

2.4 Bilevel target radius method

Consider the particular class of bilevel programming represented by problem (2.1.4). We rely on Algorithm 2 and modify its rule to define upper bounds: instead of defining $f_{k+1}^{\text{up}} = \max\{f(x_{k+1}), f_k^{\text{up}}\}$ as in Step 5 of Algorithm 2, the method given below sets upper bounds as $f_{k+1}^{\text{up}} = f(y_{k+1})$, where $\{y_k\} \subset X$ is a (second) sequence of points generated by the algorithm. More specifically, the sequence $\{y_k\}$ is generated as follows: set $y_0 = x_0$, the initial point; for $k > 0$, set $y_{k+1} = y_k$ if $f(x_{k+1}) > f(y_k) - \frac{1}{2}(1-\gamma_1)\Delta_k$, otherwise y_{k+1} is the (unique) solution of the following extra subproblem

$$\begin{cases} \min\limits_{y \in X} & \mathfrak{c}(y) \\ \text{s.t.} & f(x_j) + \langle g_j, y - x_j \rangle \leq f(x_{k+1}), \quad j \in J_k \\ & f(y_k) + \langle g_{y_k}, y - y_k \rangle \leq f(x_{k+1}) \\ & \langle \nabla \mathfrak{c}(y_k), y_k - y \rangle \leq 0\,. \end{cases} \tag{2.4.1}$$

Algorithm 3 Bilevel Target Radius Algorithm

Step 1. As Step 1 of Algorithm 2. In addition set $y_0 = x_0$.

Step 2. Define $\Delta_k = f(y_k) - f_k^{\text{low}}$.
If $\Delta_k \leq \text{Tol}$, stop and return y_k and $f(y_k)$.

Step 3. Set $f_k^{\text{lev}} = f_k^{\text{low}} + \gamma_1\, \Delta_k$ and $\sigma_k = \gamma_2 \gamma_1 \Delta_k / \Lambda_k$.
Set $\tilde{x}_k = x_k$ if $k > k(l)$, or $\tilde{x}_k = y_k$, if $k = k(l)$.

Step 4. As Step 4 of Algorithm 2.

Step 5. Compute $f(x_{k+1})$, $g_{k+1} \in \partial f(x_{k+1})$ and set $\Lambda_{k+1} = \max\{1 + \|(1, g_{k+1})\|_q\,, \Lambda_k\}$.
If $f(x_{k+1}) \leq f(y_k) - \frac{1}{2}(1-\gamma_1)\Delta_k$, set y_{k+1} as the solution of (2.4.1). Else $y_{k+1} = y_k$.

Step 6. As Step 6 of Algorithm 2.

Notice that the extra subproblem (2.4.1) is solved only at iterations providing significant decrease of f.

Lemma 2.4.1. *([50, Lemma 8]). Subproblem* (2.4.1) *is well defined for all iterations* $k = 0, 1, \ldots$

Theorem 2.4.2. *([50, Theorem 9]). Consider Algorithm 3 with* $\text{Tol} = 0$ *and assume that* X *is a compact set. Then the sequence* $\{y_k\} \subset X$ *generated by the algorithm converges to an optimal solution of* (2.1.1). *Moreover,* $\bar{y} = \lim_k y_k$ *solves problem* (2.1.4).

2.5 Concluding remarks

This work significantly improves the TRM of [153] by handling problems of the form (2.1.3) and (2.1.4); different norms $\|\cdot\|_p$ ($p \in [1, \infty]$) for defining "deep cuts"; a new rule to update the inscribed sphere's radius; and by skipping solving a LP to update lower bounds. In the setting of problem (2.1.1), we

generalize the approach of [25] and extend the TRM of [153] to deal with general continuous differentiable strongly convex functions $\mathfrak{c}$ (not necessarily Bregman-like distances) exploiting the structure of the feasible set. Furthermore, we present convergence analysis of our algorithms and provide complexity results for Algorithm 2. As shown, the given algorithms generalize the well-known level bundle methods and employ a more general scheme for yielding limited memory. Our algorithms require, however, the initial point x_0 to be the minimizer of $\mathfrak{c}$ over X. While it is straightforward to determine x_0 if $\mathfrak{c}$ is a Bregman function, it might not be the case for a more general function $\mathfrak{c}$ (c.f. Remark 2.2.3). This a downside of our proposals when dealing with problems of type (2.1.3).

Chapter 3

An inertial algorithm for DC programming

This chapter is extracted from the forthcoming publication

W. de Oliveira and M. Tcheou
An inertial algorithm for DC programming
To appear in **Set-Valued and Variational Analysis**, 2018.
DOI: 10.1007/s11228-018-0497-0

The manuscript deals with nonsmooth DC programs and proposes a non-monotone algorithmic pattern with an inertial-force procedure to compute critical points for optimization problems of this class. The inertial-fore scheme, akin to the Heavy-Ball method of Polyak, helps to prevent the given algorithms from converging to bad-quality critical points. Convergence analysis and rate of convergence are presented. Moreover, one variant of the given algorithmic pattern is assessed numerically on large-scale nonconvex and nonsmooth image denoising models in computer vision systems.

3.1 Introduction

In this work we consider nonconvex nonsmooth optimization problems of the form

$$\min_{x\in\mathbb{R}^n} f(x), \qquad \text{with} \quad f(x) := f_1(x) - f_2(x)\,, \tag{3.1.1}$$

where f_1, $f_2 : \mathbb{R}^n \to \mathbb{R}\cup\{+\infty\}$ are convex and possibly nonsmooth functions. Problems of this type are known in the literature as DC programs, with "DC" standing for *Difference-of-Convex* functions [93]. We assume throughout this manuscript that f_1 is a closed function and that $\mathcal{D}\mathrm{om}(f_1) \subset \mathcal{O} \subset \mathcal{D}\mathrm{om}(f_2)$, where $\mathcal{O}$ is an open and convex set in $\mathbb{R}^n$. This assumption allows us to encompass convex constrained DC programs in formulation (3.1.1). Indeed, notice that the first component function f_1 can be, for example, the sum of a convex function $\varphi : \mathbb{R}^n \to \mathbb{R}$ with the indicator function i_X of a convex set $X \subset \mathcal{O}$, i.e., $f_1(x) = \varphi(x) + i_X(x)$ with $i_X(x) = 0$ if $x \in X$ and $i_X(x) = +\infty$ otherwise.

DC programming forms an important sub-field of nonconvex programming and has been receiving much attention from the mathematical programming community [81, 93, 95, 101, 112, 188, 193, 196].

More general DC programs with DC constraints are investigated in [113, 155, 189, 190, 191, 201]. Some applications include production-transportation planning problems [97], location planning problems [196, Chapter 5], physical layer based security in a digital communication systems [155], chance-constrained problems [50], cluster analysis [19, 104], engineering design problems [74, 196], energy management problems [201] and others. We refer to [196, Part II] for a comprehensive presentation of several algorithms

designed for DC optimization problems. Solving globally nonsmooth programs as (3.1.1) is a challenging task, especially in the large-scale setting. We will, therefore, deal with this class of problems by employing local-solution approaches.

A well-known method for dealing with the optimization problem (3.1.1) is the *DC Algorithm* (DCA) of [193] (see also [16, 112, 194]). The classical DCA handles problem (3.1.1) by defining a sequence of trial points according to the following rule, for a given starting point $x_0 \in \mathcal{D}\text{om}(f_1)$:

$$\text{for all } k = 0, 1, 2, \ldots, \text{ compute } g_2^k \in \partial f_2(x_k) \text{ and } x_{k+1} \in \partial f_1^*(g_2^k), \tag{3.1.2}$$

where $\partial f_2(x_k)$ is the subdifferential of the convex function f_2 at point x_k (see definition in Section 3.2 below) and f_1^* is the conjugate function of f_1. It can be shown [193, Theorem 3] that every cluster point $\bar{x}$ (if any) of the sequence $\{x_k\}$ generated by rule (3.1.2) is a critical point of problem (3.1.1), i.e., $\bar{x}$ satisfies

$$\partial f_1(\bar{x}) \cap \partial f_2(\bar{x}) \neq \emptyset\,. \tag{3.1.3}$$

It follows from convexity of the component functions f_1 and f_2 that rule (3.1.2) yields a monotone sequence of function values, i.e., $f(x_{k+1}) \leq f(x_k)$ for all $k = 0, 1, \ldots$ (see [193, Theorem 3(i)] for more details). While monotonicity might be seen as a quality of the method, demanding monotonically decreasing function values might not be an ideal scheme in nonconvex optimization: depending on starting points, iterates are attracted by poor-quality critical points that prevent the (monotone) algorithm from computing a critical point of better quality. In the context of DC programming, we mean by a "critical point of better quality" a critical point $\bar{x}$ that is d(irectional)-stationary, i.e., $\bar{x}$ satisfies

$$\partial f_2(\bar{x}) \subset \partial f_1(\bar{x})\,. \tag{3.1.4}$$

As shown in [93, 155], d-stationarity is the sharpest stationary definition for nonconvex problems of type (3.1.1); see also additional comments in Section 3.2. It is clear from its definition that computing d-stationary points for (3.1.1) is not a trivial task in general. Few exceptions are the situations in which either f_1 or f_2 are differentiable, [73, Section 2], and the case when f_2 is the pointwise maximum of finitely many convex and differentiable functions. The latter case is investigated in [155], where the authors propose a proximal linearized method that solves several convex programs per iteration, and thus has a high computational burden. A less computational demanding method is a variant of the proximal bundle algorithm proposed in [50], but may require solving many quadratic programs per iteration; see [50, Algorithm 2].

In this work, we are concerned with algorithms of low computational costs to compute critical points. Moreover, we do not assume that f_2 is the pointwise maximum of finitely many convex and differentiable functions. In order to try to transpose critical points that are not d-stationary, we furnish the DCA algorithm represented by rule (3.1.2) with an inertial scheme that can be seen as a version of the heavy-ball method of Polyak [162].

In a differentiable and convex framework, the heavy-ball method is a two-step gradient algorithm that can be interpreted as an explicit finite differences discretization of the so-called *heavy-ball with friction dynamical system* [146]. In summary, the algorithm incorporates an inertial force in the iterative process of gradient methods by appending to the negative-gradient direction the inertial term $\gamma(x_k - x_{k-1})$, where $\gamma \geq 0$ is a given parameter. The heavy-ball method has been generalized in several manners: the authors of [216] consider differentiable but nonconvex optimization problems, and in [14, 15, 139] the heavy-ball method was extended to handle maximal monotone operators. The paper [146] deals with nonsmooth nonconvex optimization problems of the form $\min_{x \in \mathbb{R}^n} f_1(x) + \zeta(x)$, where f_1 is as above and $\zeta : \mathbb{R}^n \to \mathbb{R}$ is a smooth nonconvex function with Lipschitz continuous gradient. The authors of [146] assume $f_1 + \zeta$ to be coercive and propose a linearized proximal method with inertial force to compute stationary points.

In the DC context, $\zeta = -f_2$ is a concave function. However, in this work we do not assume that either f_2 is smooth nor $f = f_1 - f_2$ is coercive. Our proposal follows the general lines of the DCA and replaces the subgradient g_2^k of the second component function f_2 in rule (3.1.2) by $g_2^k + \gamma(x_k - x_{k-1})$, with $\gamma \geq 0$.

This leads to the following iterative scheme, for a given point $x_0 \in \mathcal{D}\text{om}(f_1)$:

$$\text{for all } k = 0, 1, 2, \ldots, \text{ compute } g_2^k \in \partial f_2(x_k) \text{ and } x_{k+1} \in \partial f_1^*(g_2^k + \gamma(x_k - x_{k-1})). \tag{3.1.5}$$

As we will see in Section 3.4, the sequence of function values issued by the above scheme is not necessarily monotone due to the inertia imposed by the term $\gamma(x_k - x_{k-1})$. As already said, this property can be beneficial for the quest of computing critical points that are also d-stationary for (3.1.1). Nevertheless, it is not assured that our *Inertial DC Algorithm* (InDCA) illustrated by rule (3.1.5) will always compute d-stationary points, but critical ones. Numerical experiments reported in Section 3.6 below show that InDCA is more robust than DCA, meaning that for the same starting points InDCA computes very often better critical points than DCA does.

As for (3.1.2), rule (3.1.5) is not practical when a subgradient of f_1^* is not easily computed. For this reason, the given algorithm relaxes the requirement $x_{k+1} \in \partial f_1^*(g_2^k + \gamma(x_k - x_{k-1}))$ by a milder one, that can be understood as demanding x_{k+1} to belong to an approximate subdifferential of f_1^* at point $g_2^k + \gamma(x_k - x_{k-1})$ with a vanishing approximation error. (This is done by inexactly solving the primal subproblem given in (3.2.2) below). This is the second property of practical interest of our algorithm as it substantially reduces the computational burden to compute a d-stationary/critical point of DC programs.

The remainder of this work is organized as follows: Section 3.2 provides some notation and preliminary results that will be employed throughout the text. In addition, the section contains two examples illustrating the benefits of incorporating an inertial force to the DC algorithm. Section 3.3 presents our inertial DC algorithmic pattern as well as some practical issues concerning the implementation of some of its variants. Convergence analysis and convergence rate are considered in Section 3.4, and the application of interest is discussed in Section 3.5: nonconvex image denoising models. Section 3.6 reports some preliminary numerical results comparing InDCA against DCA and the nonconvex algorithm iPiano of [146]. Finally, Section 3.7 closes the paper with some concluding remarks.

3.2 Notation, main definitions and illustrative examples

For any points $x, y \in \mathbb{R}^n$, $\langle x, y\rangle$ stands for the Euclidean inner product, and $\|\cdot\|$ for the associated norm, i.e., $\|x\| = \sqrt{\langle x, x\rangle}$. For a set $X \subset \mathbb{R}^n$, we denote by i_X its indicator function, i.e., $i_X(x) = 0$ if $x \in X$ and $i_X(x) = +\infty$ otherwise. For a convex set X its normal cone at the point x is denoted by $N_X(x)$, which is the set $\{y : \langle y, z - x\rangle \leqslant 0 \, \forall z \in X\}$ if $x \in X$ and the empty set otherwise.

As convex functions are directionally differentiable in the interior of their domains [173, Theorems 23.1 and 23.4], the limit

$$f_i'(x; d) := \lim_{t \downarrow 0} \frac{f_i(x + td) - f_i(x)}{t}$$

for f_i, $i = 1, 2$, is well defined for all x in the interior of $\mathcal{D}\text{om}(f_i)$ and all $d \in \mathbb{R}^n$. It is well known that $f_i'(x; d) = \max_{g \in \partial f_i(x)} \langle g, d\rangle$, where

$$\partial f_i(x) := \{g \in \mathbb{R}^n : \, f_i(y) \geq f_i(x) + \langle g, y - x\rangle \quad \forall y \in \mathbb{R}^n\}$$

is the subdiferential of f_i at point x. For $\epsilon \geq 0$ the ϵ-subdiferential is denoted by

$$\partial_\epsilon f_i(x) := \{g \in \mathbb{R}^n : \, f_i(y) \geq f_i(x) + \langle g, y - x\rangle - \epsilon \quad \forall y \in \mathbb{R}^n\}.$$

Since DC functions are also locally Lipschitz continuous (because their components f_i are so), their directional derivatives are well defined for all x in the interior of $\mathcal{D}\text{om}(f_i)$, $i = 1, 2$:

$$f'(x; d) = f_1'(x; d) - f_2'(x; d)\,.$$

A point $\bar{x} \in \mathbb{R}^n$ is a d(irectional)-stationary point of problem (3.1.1) if $f'(\bar{x}; (x - \bar{x})) \geq 0$ for all $x \in \mathbb{R}^n$, which can be shown to be equivalent to the inclusion (3.1.4), [93]. Notice that verifying

(3.1.4) computationally is impractical in many cases of interest. Hence, one generally employs a weaker notion of stationarity: a point $\bar{x} \in \mathbb{R}^n$ is called a *critical point* of problem (3.1.1) if $\bar{x}$ satisfies (3.1.3). In summary, all local minimizers of problem (3.1.1) are d-stationary points, which in turn are critical points of (3.1.1). The reverse implications are, in general, not true as illustrated in [155, Example 2] (see also Example 3.2.1 below).

In what follows we consider rules (3.1.2) and (3.1.5) and present practical manners to define trial points. We recall that the conjugate of a convex function f_i^* is

$$f_i^*(g_i) := \sup_{x \in \mathbb{R}^n} \{\langle g_i, x\rangle - f_i(x)\}.$$

Since f_1 is a closed and convex function, we have that $x_{k+1} \in \partial f_1^*(g_2^k)$ (rule (3.1.2)) if and only if $g_2^k \in \partial f_1(x_{k+1})$ [96, Prop. 6.1.2]. The latter inclusion is satisfied if x_{k+1} is a solution of the following convex subproblem

$$\min_{x \in \mathbb{R}^n} f_1(x) - \langle g_2^k, x\rangle. \tag{3.2.1}$$

Analogously, the condition $x_{k+1} \in \partial f_1^*(g_2^k + \gamma(x_k - x_{k-1}))$ of (3.1.5) is satisfied if x_{k+1} solves

$$\min_{x \in \mathbb{R}^n} f_1(x) - \langle g_2^k + \gamma(x_k - x_{k-1}), x\rangle. \tag{3.2.2}$$

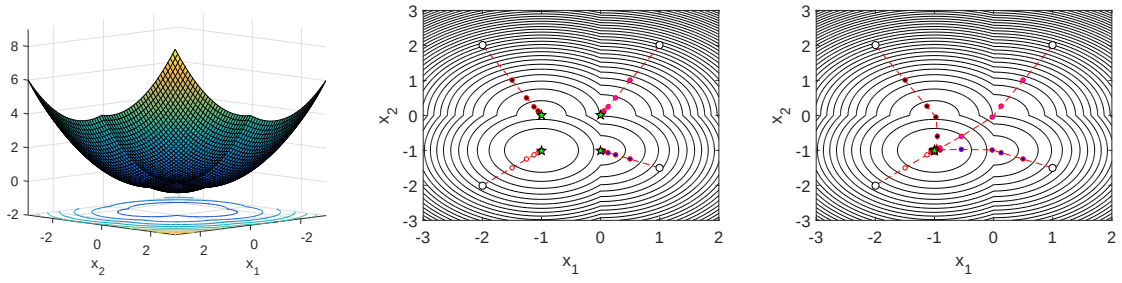
Throughout this work we assume the following condition, which is a mild hypothesis in the DC setting:
Assumption A1. *Function f_2 is strongly convex on $\mathcal{O}$ with a* known *parameter $\rho > 0$, that is, for every $g_2 \in \partial f_2(x)$ one has*

$$f_2(y) \geq f_2(x) + \langle g_2, y - x\rangle + \frac{\rho}{2}\|y - x\|^2, \quad \forall\, x, y \in \mathcal{O}. \tag{3.2.3}$$

We care to mention that A1, also present in [201], is not a restrictive assumption at all. In fact, if A1 does not hold for a certain DC function $f = \varphi - \psi$ we can obtain another DC decomposition of f satisfying A1 by adding an arbitrary strongly convex function $\mathfrak{s} : \mathcal{O} \to \mathbb{R}$ to the component functions: note that $f = f_1 - f_2$ with $f_1 = \varphi + \mathfrak{s}$ and $f_2 = \psi + \mathfrak{s}$. Since one can always take $\mathfrak{s}(\cdot) = \|\cdot\|^2$, then ρ can be assumed known in A1 without loss of generality (just take $\rho = 2$ in this case).

Under A1 and the hypothesis that the inertial parameter γ in (3.1.5) satisfies $0 \leq \gamma < \rho/2$ we illustrate the behavior of the sequences generated by rules (3.1.2) and (3.1.5) in the following two examples.

Example 3.2.1 (Transposing critical points that are not d-stationary)**.** *Consider the bi-dimensional DC function $f(x) = f_1(x) - f_2(x)$, with $f_1(x) = \|x\|^2$ and $f_2(x) = \max(-x_1, 0) + \max(-x_2, 0) + 0.5\|x\|^2$. Its curve is plotted in Figure 3.2.1, as well as the behavior of some sequences of points generated by defining the next iterate x_{k+1} as in* (3.1.2) *(DCA, Figure 3.2.1(b)) and sequences generated by the new rule* (3.1.5) *(InDCA, Figure 3.2.1(c)) with inertial factor $\gamma = 0.49$, which was chosen to be less than $\rho/2 = 0.5$.*



(a) DC function (b) DCA (c) InDCA with $\gamma = 0.49$

Figure 3.1: Iterative process and level curves of function $f(x) = f_1(x) - f_2(x)$, with $f_1(x) = \|x\|^2$ and $f_2(x) = \max(-x_1, 0) + \max(-x_2, 0) + 0.5\|x\|^2$. Comparison between the classical DCA and the proposed inertial DC algorithm.

For the four different starting points, DCA determined four different critical points, all presented in Table 3.2.1. However, the global solution $\bar{x} = (-1,-1)^\top$ is the only d-stationary point of the problem of minimizing f over $\mathbb{R}^2$. □

$\bar{x}$	$f(\bar{x})$	$\partial f_1(\bar{x})$	$\partial f_2(\bar{x})$
$(-1,-1)^\top$	-1	$(-2,-2)^\top$	$(-2,-2)^\top$
$(0,-1)^\top$	-0.5	$(0,-2)^\top$	$(s,-2)^\top$ with $s \in [-1,0]$
$(-1,0)^\top$	-0.5	$(-2,0)^\top$	$(-2,s)^\top$ with $s \in [-1,0]$
$(0,0)^\top$	0	$(0,0)^\top$	$(s_1,s_2)^\top$ with $s_1, s_2 \in [-1,0]$

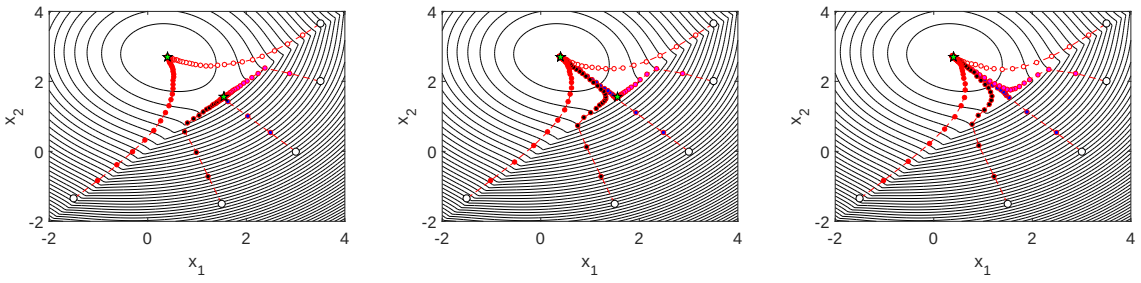
Table 3.1: Critical points of function $f(x) = \|x\|^2 - [\max(-x_1,0) + \max(-x_2,0) + 0.5\|x\|^2]$ determined by rule (3.1.2).

While the critical points computed by the classical DC algorithm depend strongly on the starting points, the inertial DC algorithm is able (in this example) to compute the d-stationary point (in this case a global solution) regardless the initial point. This is thanks to the inertial factor $\gamma(x_k - x_{k-1})$ that prevents the iterative process from stopping at critical points that are not d-stationary. In some situations it is also possible to overcome local solutions, as illustrated by the following example.

Example 3.2.2 (Trasposing local minimizers: a two-variable nonconvex 1D denoising model)**.** *We consider the following nonconvex denoising optimization problem for 1D signals:*

$$\min_{x\in\mathbb{R}^n} \frac{\mu}{2}\|x-b\|^2 + \sum_{i=1}^{n-1} \phi(|x_{i+1}-x_i|)\,.$$

The concave function $\phi(r) := \log(1+2r)/2$ is employed to induce sparsity of the one-lag-difference of the reconstructed signal $\bar{x}$: one wishes to reconstruct piecewise constant signals. As it will be shown later (see Proposition 3.5.2) the above nonconvex objective is indeed a DC function $f = f_1 - f_2$, with possible DC components given by $f_1(x) := \frac{\mu}{2}\|x-b\|^2 + \sum_{i=1}^{n-1}|x_{i+1}-x_i| + \|x\|^2$ and $f_2(x) := \sum_{i=1}^{n-1}|x_{i+1}-x_i| - \sum_{i=1}^{n-1}\phi(|x_{i+1}-x_i|) + \|x\|^2$ (hence the parameter of strongly convexity of f_2 is $\rho \geq 2$). In order to analyze the iterative process yielded by rules (3.1.2) and (3.1.5) applied to this problem, we consider dimension $n = 2$ and parameters $\mu = 0.6$, $b = (0.1, 3)^\top$. Notice that differently from Example 3.2.1, subproblems (3.1.2) and (3.1.5) do not have explicit solutions. We therefore compute iterates by solving these subproblems numerically up to a given tolerance. The iterative processes with five different initial points are presented in Figure 3.2.2, with $0 \leq \gamma < \rho/2$. In three of the five initial points,



(a) DCA (b) InDCA $\gamma = 0.7$ (c) InDCA $\gamma = 0.9$

Figure 3.2: Iterative process and level curves of function $f(x) = \frac{\mu}{2}\|x-b\|^2 + \sum_{i=1}^{n-1}\phi(|x_{i+1}-x_i|)$, with $\mu = 0.6$, $b = (0.1, 3)^\top$ and $\phi(r) = \log(1+2r)/2$. The global solution is $\bar{x} = (0.3970, 2.7030)^\top$ and the optimal value is $f(\bar{x}) \approx 0.91538$. The set of critical points of f that does not contain the global solution is $C = \{(r,r) : \max\{b_1,b_2\} - 1/\mu \leq r \leq \min\{b_1,b_2\} + 1/\mu\}$, i.e, $C \approx \{(r,r) : 1.333 \leq r \leq 1.7667\}$.

sequences generated by DCA could not converge to the global minimum $\bar{x} = (0.3970, 2.7030)^\top$, but to the critical point $(31/20, 31/20)^\top$ that is a local solution (thus a d-stationary point) of the problem. Instead,

rule (3.1.5) *was more successful due to inertial factor* γ*: for* $\gamma = 0.7$*, only one sequence converged to the critical point that is not the global minimum. Yet, for* $\gamma = 0.9$ *all five sequences converged to the global solution.* □

Example 3.2.2 suggests to consider a larger factor of inertia $\gamma \geq 0$. However, our analysis given in Section 3.3 shows that γ cannot be arbitrarily large: the inertial parameter can vary along the interval $[0, \rho/2)$, where $\rho \geq 0$ is the constant of strongly convexity of the second component function f_2.

3.3 An inertial DC algorithmic pattern

In this section, we formalize our inertial DC algorithm (InDCA) represented by rule (3.1.5) (which is related to iteratively computing a solution of (3.2.2)). We care to mention that if subproblem (3.2.2) is difficult to solve (e.g. when f_1 is assessed via simulation, optimization, numerical multidimensional integration etc.) then defining trial points by rule (3.1.5) (as well as (3.1.2)) can be too time consuming. To overcome this difficulty we follow the lead of [188, 191] and allow trial points to be inexact solutions of subproblem (3.2.2). In [191] trial points are defined as ϵ^{k+1}-solutions of the convex subproblems, where $\epsilon^{k+1} \to 0$. This idea is also explored in the context of linearized proximal methods in [188].

Differently from [188, 191] we define the trial point x_{k+1} in such a manner that the ϵ^{k+1}-subdifferential of f_1 at x_{k+1} intersects the set $\partial f_2(x_k) + \gamma(x_k - x_{k-1})$:

$$\partial_{\epsilon^{k+1}} f_1(x_{k+1}) \cap \partial f_2(x_k) + \gamma(x_k - x_{k-1}) \neq \emptyset\,.$$

The motivation for such a strategy lies in the fact that if the sequence $\{x_k\}$ converges to a point $\bar{x}$ and $\{\epsilon^k\}$ vanishes, then the above condition eventually implies criticality (3.1.3) of $\bar{x}$. In fact, as it will be shown in Section 3.4, convergence of the whole sequence $\{x_k\}$ is not required: any cluster point of $\{x_k\}$ can be shown to be a critical point of (3.1.1). Furthermore, the inexactness ϵ^{k+1} involved in the iterative process is automatically controlled by our algorithm in such a manner that errors vanish as the iterative process progresses. The InDCA is presented in the following algorithmic pattern.

Algorithm 4 INERTIAL DC ALGORITHMIC PATTERN

Let $x_0 \in \mathcal{D}\mathrm{om}(f_1)$, Tol ≥ 0, $\lambda \in [0, 1)$ and $\gamma \in [0, (1-\lambda)\rho/2)$ be given. Set $x_{-1} = x_0$
for $k = 0, 1, 2, \ldots$ **do**
  Set $d^k = \gamma(x_k - x_{k-1})$ and find $x_{k+1} \in \mathbb{R}^n$ such that

$$\partial_{\epsilon^{k+1}} f_1(x_{k+1}) \cap \partial f_2(x_k) + d^k \neq \emptyset \quad \text{with} \quad 0 \leq \epsilon^{k+1} \leq \lambda \frac{\rho}{2} \|x_{k+1} - x_k\|^2 \tag{3.3.1}$$

  if $\|x_{k+1} - x_k\| \leq$ Tol and $\|d^k\| \leq$ Tol **then**
    Stop and return $(x_k, f(x_k))$
  end if
end for

The following is an alternative to condition (3.3.1) that is suitable when the first component function f_1 is smooth: compute $x_{k+1} \in \mathbb{R}^n$ such that

$$\left\|g_1^{k+1} - (g_2^k + d^k)\right\| \leq \lambda \frac{\rho}{2} \|x_{k+1} - x_k\| \quad \text{for some} \quad g_1^{k+1} \in \partial f_1(x_{k+1}) \text{ and } g_2^k \in \partial f_2(x_k)\,. \tag{3.3.2}$$

The Algorithmic pattern 4 boils down to specific optimization algorithms upon the choice of its parameters. We start by addressing some particular cases issued by the choice $\lambda = 0$. The choice $\lambda > 0$ means that subproblem (3.1.5) can be inexactly solved. Practical details on how to implement (3.3.1) and (3.3.2) for $\lambda > 0$ are given in Subsection 3.3.2.

3.3.1 Some specific settings for the algorithmic pattern with $\lambda = 0$

3.3.1.1 The DC algorithm with/without inertial force

Consider the Algorithmic pattern 4 with $\lambda = \gamma = 0$. With this choice of parameters condition (3.3.1) is equivalent to (3.3.2), which reads as

$$\partial f_1(x_{k+1}) \cap \partial f_2(x_k) \neq \emptyset\,.$$

Such condition is ensured, for instance, if $g_2^k \in \partial f_2(x_k)$ and x_{k+1} solves $\min_{x\in\mathbb{R}^n} f_1(x) - \langle g_2^k, x\rangle$, the subproblem of the classic DC algorithm of [193]: optimality of x_{k+1} implies $g_2^k \in \partial f_1(x_{k+1})$, and therefore $g_2^k \in \partial f_1(x_{k+1}) \cap \partial f_2(x_k)$.

If $\lambda = 0$ but $\gamma > 0$ then either (3.3.1) or (3.3.2) is equivalent to define x_{k+1} by solving (3.2.2): its optimality condition is $g_2^k + \gamma(x_k - x_k) \in \partial f_1(x_{k+1})$. We use this property in the sequel.

3.3.1.2 The linearized proximal method with/without inertial force

Consider $\lambda = 0$ in the Algorithmic pattern 4. Suppose that $\mathfrak{s} : \mathbb{R}^n \to \mathbb{R}_+$ is a strongly convex and continuously differentiable function, and that $f_1(x) = \varphi(x) + \mathfrak{s}(x)$ and $f_2(x) = \psi(x) + \mathfrak{s}(x)$, i.e., $\mathfrak{s}$ is a regularizing function. Under this assumption, $g_2^k \in \partial f_2(x_k)$ is given by $g_2^k = g_\psi^k + \nabla\mathfrak{s}(x_k)$, with $g_\psi^k \in \partial\psi(x_k)$. Accordingly, subproblem (3.2.2) (yielding (3.3.1) when $\lambda = 0$) becomes

$$\min_{x\in\mathbb{R}^n} \varphi(x) + \mathfrak{s}(x) - \langle g_\psi^k + \nabla\mathfrak{s}(x_k) + \gamma(x_k - x_{k-1}), x\rangle\,.$$

By adding the constant term $-\mathfrak{s}(x_k) + \langle g_\psi^k + \nabla\mathfrak{s}(x_k) + \gamma(x_k - x_{k-1}), x_k\rangle$ to the above subproblem we get

$$\begin{aligned}\min_{x\in\mathbb{R}^n} \varphi(x) + \mathfrak{s}(x) - \mathfrak{s}(x_k) - \langle g_\psi^k + \nabla\mathfrak{s}(x_k) + \gamma(x_k - x_{k-1}), x - x_k\rangle \\ = \min_{x\in\mathbb{R}^n} \varphi(x) - \langle g_\psi^k + \gamma(x_k - x_{k-1}), x - x_k\rangle + D(x, x_k),\end{aligned} \tag{3.3.3}$$

with $D(x, x_k) := \mathfrak{s}(x) - \mathfrak{s}(x_k) - \langle\nabla\mathfrak{s}(x_k), x - x_k\rangle$ the Bregman function induced by $\mathfrak{s}$. Hence, Algorithm 4 becomes an *Inertial Linearized Proximal Method* for DC programming, a new variant of proximal methods.

In particular, suppose that $\mathfrak{s}(x) = \frac{\rho}{2}\|x\|^2$. Then the solution of subproblem (3.3.3) also solves

$$\min_{x\in\mathbb{R}^n} \varphi(x) - \langle g_\psi^k, x\rangle + \frac{\rho}{2}\left\|x - \left[x_k + \frac{\gamma}{\rho}(x_k - x_{k-1})\right]\right\|^2. \tag{3.3.4}$$

We care to mention that depending on the structure of φ, subproblem (3.3.4) can be efficiently solved by specialized algorithms. This is the case of nonconvex image denoising models, the application considered in Sections 3.5 and 3.6.

If one chooses $\gamma = 0$, then Algorithm 4 satisfying (3.3.1) with $\lambda = 0$ by solving (3.3.4) is just the linearized proximal method applied to problem (3.1.1) [155, 188]. On the other hand, if $\gamma > 0$ then Algorithm 4 can be seen as an extension of the *iPiano algorithm* of [146] to deal with nonsmooth DC programs (the original algorithm of [146] requires the nonconvex function -ψ to be differentiable and its gradient to be Lipschitz continuous, an assumption that is not required here).

3.3.1.3 Convex setting: proximal method and proximal subgradient splitting method

Once again, suppose that f in (3.1.1) is given by $f_1(x) = \varphi(x) + \frac{\rho}{2}\|x\|^2$ and $f_2(x) = \psi(x) + \frac{\rho}{2}\|x\|^2$. As seen above, subproblem (3.2.2) becomes (3.3.4). Furthermore, suppose that ψ is not convex but a concave function. Then Algorithm 4 satisfying (3.3.1) with $\lambda = 0$ by solving (3.3.4) becomes a *proximal*

subgradient splitting method [22] applied to the (now convex) problem $\min_{x\in\mathbb{R}^n} \varphi(x) - \psi(x)$. Differently from the proximal subgradient splitting methods found in the literature employing (3.3.4) with $\gamma = 0$, the one resulting from Algorithm 4 (under the above assumptions) is of the inertial type because it allows $\gamma \neq 0$. Besides, if $\psi := 0$ then subproblem (3.3.4) yields an inertial iteration of a *proximal method* applied to (3.1.1), which is (in this particular case) simply $\min_{x\in\mathbb{R}^n} \varphi(x)$.

3.3.2 Some specific settings for the algorithmic pattern with $\lambda > 0$

We start by showing that if $\lambda > 0$, then Algorithm 4 relates to the local-search method of [191].

3.3.2.1 A local-search like method with/without inertial force

Note that (3.3.1) implies the following inclusion, where g_2^k is a subgradient of f_2 at point x_k,

$$g_2^k + \gamma(x_k - x_{k-1}) \in \partial_{\epsilon^{k+1}} f_1(x_{k+1}).$$

The definition of the ϵ^{k+1}-subdifferential $\partial_{\epsilon^{k+1}} f_1(x_{k+1})$ provides the inequality

$$f_1(x) \geq f_1(x_{k+1}) + \langle g_2^k + \gamma(x_k - x_{k-1}), x - x_{k+1}\rangle - \epsilon^{k+1} \quad \text{for all } \; x \in \mathbb{R}^n,$$

which in turn gives

$$f_1(x) - \langle g_2^k + \gamma(x_k - x_{k-1}), x\rangle \geq f_1(x_{k+1}) - \langle g_2^k + \gamma(x_k - x_{k-1}), x_{k+1}\rangle - \epsilon^{k+1} \quad \text{for all } \; x \in \mathbb{R}^n.$$

In particular, x_{k+1} is an ϵ^{k+1}-solution of (3.2.2). This procedure is akin to the local-search method of [191], but with the following differences: in [191] $\gamma = 0$ and the error ϵ^{k+1} must be chosen to form a summable series. We care to mention that the local-search of [191] is general enough to handle DC constraints, which is not the case of Algorithm 4. An extension of our algorithm to handle DC constraints is left for future investigation.

3.3.2.2 Bundle-like algorithm with/without inertial force

Suppose that $\mathfrak{s} : \mathbb{R}^n \to \mathbb{R}_+$ is a strongly convex and continuously differentiable function, $f_1(x) = \varphi(x) + i_X(x) + \mathfrak{s}(x)$, $f_2(x) = \psi(x) + \mathfrak{s}(x)$ and X is a convex set. In order to compute a point x_{k+1} satisfying (3.3.1) we consider subproblem (3.2.2), which reads as

$$\min_{x\in X} \varphi(x) + \mathfrak{s}(x) - \langle g_2^k + \gamma(x_k - x_{k-1}), x\rangle\,, \tag{3.3.5}$$

and an "inner" iterative process $\nu = 0, 1, 2, \ldots$ generating a sequence of iterates $\{z^\nu\}$ whose clusters points solve (3.3.5). Instead of defining x_{k+1} as one of the cluster points of $\{z^\nu\}$, we break the inner iterative process and define $x_{k+1} := z^{\nu+1}$ as soon as $z^{\nu+1}$ is an $\epsilon^{\nu+1}$-solution of (3.3.5), with $\epsilon^{\nu+1}$ satisfying the right-side of (3.3.1). To this end, we replace the convex function φ in (3.3.5) with a cutting-plane model $\check{\varphi}^\nu$ defined by

$$\check{\varphi}^\nu(x) := \max_{j\leq\nu}\{\varphi(z^j) + \langle g_\varphi^j, x - z^j\rangle\} \quad \text{with} \quad g_\varphi^j \in \partial\varphi(z^j),\; j = 0, 1, \ldots, \nu\,. \tag{3.3.6}$$

Convexity of φ ensures that φ^ν approximates φ from below: $\varphi^\nu(x) \leq \varphi(x)$ for all x. These are the main ingredients of the following implementable scheme for computing a trial point x_{k+1} satisfying (3.3.1).

Proposition 3.3.1. *([51, Proposition 1]). Let k be fixed and suppose that x_k does not solve* (3.3.5). *Then Algorithm 5 stops after finitely many steps ν with a point $x_{k+1} := z^{\nu+1}$ satisfying* (3.3.1).

If x_k is the unique solution of (3.3.5), then Algorithm 5 ensures (under the analysis of [43]) that $\{z^\nu\}$ converges to x_k. Therefore, to guarantee that Algorithm 5 will always halt after finitely many steps one may consider the following additional test ensuring that $z^{\nu+1}$ is a Tol-solution of (3.3.5): if $\varphi(z^{\nu+1}) - \varphi^\nu(z^{\nu+1}) \leq \text{Tol}$, then exit $x_{k+1} := z^{\nu+1}$.

Algorithm 5 An implementable scheme for satisfying (3.3.1)

1: Given $\lambda > 0$, $0 \leq \gamma < \rho/2$, x_k and x_{k-1}, set $d^k = \gamma(x_k - x_{k-1})$ and $z^0 = x_k$
2: Compute $(\varphi(z^0),\, g_\varphi^0 \in \partial\varphi(z^0))$

3: **for** $\nu = 0, 1, 2, \ldots$ **do**
4: Define $\check{\varphi}^\nu$ as in (3.3.6) and let $z^{\nu+1}$ be a solution of

$$\min_{x \in X} \check{\varphi}^\nu(x) + \mathfrak{s}(x) - \langle g_2^k + d^k, x\rangle \tag{3.3.7}$$

5: Compute $(\varphi(z^{\nu+1}),\, g_\varphi^{\nu+1} \in \partial\varphi(z^{\nu+1}))$ and set $\epsilon^{\nu+1} := \varphi(z^{\nu+1}) - \check{\varphi}^\nu(z^{\nu+1})$
6: **if** $\epsilon^{\nu+1} \leq \lambda\frac{\rho}{2}\left\|z^{\nu+1} - x_k\right\|^2$ **then**
7: Stop and exit with $x_{k+1} := z^{\nu+1}$
8: **end if**
9: **end for**

Computing a point satisfying condition (3.3.2)

We now show how to compute x_{k+1} satisfying (3.3.2) for a particular class of problem (3.1.1) whose first component f_1 is of class C^1 and the domains of both f_1 and f_2 is the whole space $\mathbb{R}^n$. Given an arbitrary vector $g_2^k \in \partial f_2(x_k)$, let $y(x_k) \in \mathbb{R}^n$ be a solution of subproblem (3.1.5). Under the given assumptions, it follows that $\nabla f_1(y(x_k)) - (g_2^k + d^k) = 0$. Let $\{z^\nu\}$ be a sequence of points generated by a convergent algorithm applied to (3.1.5) (e.g. a Newtonian method, [100]) such that $\lim_{\nu \in N'} z^{\nu+1} = y(x_k)$. Then, by continuity of ∇f_1 we conclude that

$$\lim_{\nu \in N'} \nabla f_1(z^{\nu+1}) = \nabla f_1(y(x_k)) = g_2^k + d^k.$$

If there is no index $\nu \in N'$ such that $\left\|\nabla f_1(z^{\nu+1}) - (g_2^k + d^k)\right\| \leq \lambda\frac{\rho}{2}\left\|z^{\nu+1} - x_k\right\|$, then $\{z^{\nu+1}\}_{N'}$ would converge to x_k faster than $\{\nabla f_1(z^{\nu+1})\}_{N'}$ converges to $g_2^k + d^k$. This yields $y(x_k) = x_k$ and $\nabla f_1(x_k) = g_2^k + d^k$, proving that x_k is a critical point of (3.1.1) if $d^k = 0$. Otherwise, Algorithm 4 sets $x_{k+1} = x_k$ (resulting $d^{k+1} = 0$) and proceeds to next iteration. (If $d^k = 0$ and x_k is not a critical point, the condition of (3.3.2) can be satisfied after finitely many steps by some inner iterate $z^{\nu+1}$.)

3.3.3 An alternative stopping test

The stopping test given in the Algorithmic pattern 4 is a reliable and straightforward one. However, it may not scale well the underlying optimization problem when the function is "flat" around a critical point and/or when the dimension of x is very large.

A more practical stopping test depending on the function values has been investigated in [191, Remark 4]. We thus rely on [191] and Lemma 3.4.2 below to propose the following alternative stopping test for Algorithm 4:

An alternative stopping test for Algorithm 4
1: **if** $\left|f(x_{k+1}) + \frac{(1-\lambda)\rho-\gamma}{2}\left\|x_{k+1} - x_k\right\|^2 - \left[f(x_k) + \frac{(1-\lambda)\rho-\gamma}{2}\left\|x_k - x_{k-1}\right\|^2\right]\right| \leq$ Tol **then**
2: Stop and return $(x_k, f(x_k))$
3: **end if**

As it will be seen in Lemma 3.4.2, the sequence $\{f(x_{k+1}) + \frac{(1-\lambda)\rho-\gamma}{2}\left\|x_{k+1} - x_k\right\|^2\}$ is monotonically decreasing. The reasoning of the above test is to stop the algorithm when the decrease issued by such a sequence is small enough.

3.4 Convergence analysis

As point x_{k+1} is chosen to satisfy either (3.3.1) or (3.3.2) we conclude that $\partial f_1(x_{k+1}) \neq \emptyset$ and therefore $x_{k+1} \in \mathcal{D}\text{om}(f_1)$, implying that $\{x_k\} \subset \mathcal{D}\text{om}(f_1)$. By assumption, there exists an open set in $\mathbb{R}^n$ satisfying $\mathcal{D}\text{om}(f_1) \subset \mathcal{O} \subset \mathcal{D}\text{om}(f_2)$. Hence, the sequence of points generated by Algorithm 4 is well defined. Furthermore, since f_2 is a convex function, its subdifferential ∂f_2 is locally bounded. As a result, any sequence $\{g^k\}$ with $g^k \in \partial_{\epsilon^{k+1}} f_1(x_{k+1}) \cap \partial f_2(x_k) + d^k$ is bounded as long as $\{x_k\}$ is bounded as well. With this in mind, we start the convergence analysis of Algorithm 4 with the following simple lemma. Throughout this subsection we consider Algorithm 4 with $\text{Tol} = 0$.

Lemma 3.4.1. *([51, Lemma 1]). Suppose Algorithm 4 terminates at iteration k. Then x_k is critical point of problem* (3.1.1).

The following lemma plays an important role in the convergence analysis of Algorithm 4.

Lemma 3.4.2. *([51, Lemma 2]). Let $\{x_k\}$ be the sequence generated by Algorithm 4. If assumption A1 holds, $\lambda \in [0, 1)$ and $\gamma \in [0, (1-\lambda)\rho/2)$, then $\frac{(1-\lambda)\rho-2\gamma}{2} > 0$ and the sequence $\{f(x_k) + \frac{(1-\lambda)\rho-\gamma}{2}\|x_k - x_{k-1}\|^2\}$ is monotonically decreasing, that is*

$$\begin{aligned} f(x_{k+1}) + \frac{(1-\lambda)\rho-\gamma}{2}\|x_{k+1}-x_k\|^2 \quad &\leq \quad f(x_k) + \frac{(1-\lambda)\rho-\gamma}{2}\|x_k - x_{k-1}\|^2 - \\ &\qquad \frac{(1-\lambda)\rho-2\gamma}{2}\|x_k - x_{k-1}\|^2 \quad \text{for all } k = 0, 1, 2, \ldots \end{aligned}$$

The property that the sequence $\{f(x_k) + \frac{(1-\lambda)\rho-\gamma}{2}\|x_k - x_{k-1}\|^2\}$ is monotonically decreasing is enough to prove convergence of Algorithm 4. We care to mention that the sequence of functional value $\{f(x_k)\}$ is not necessary monotone, in contrast to all other DC algorithms found in the literature (see for instance, [50, 73, 101, 155, 188, 193]). Remind that the non-monotonicity of the function values was crucial for InDCA to escape from the local solution in Example 3.2.2.

Theorem 3.4.3. *([51, Theorem 1]). Consider Algorithm 4, assume A1, $\lambda \in [0, 1)$, $\gamma \in [0, (1-\lambda)\rho/2)$, and $x_0 \in \mathcal{D}\text{om}(f_1)$. Assume also that the level set $L_f(x_0) := \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is bounded. Then any cluster point $\bar{x}$ of the sequence $\{x_k\}$ generated by the algorithm is a critical point of problem* (3.1.1).

Once the convergence analysis of Algorithm 4 has been established, we now turn our attention to its speed of convergence. To this end, we consider a particular instance of Algorithm 4.

3.4.1 Rate of convergence

Throughout this section, we assume that $f_1(x) = \varphi(x) + \|x\|^2/2$ and $f_2(x) = \psi(x) + \|x\|^2/2$, yielding $\rho \geq 1$ in A1 and $\gamma \in [0, 1/2)$ in Algorithm 4. We moreover assume that $\lambda = 0$ in condition (3.3.1) and let g_ψ^k be an arbitrary subgradient of ψ at point x_k. In this manner, determining x_{k+1} satisfying either (3.3.1) or (3.3.2) results in defining (for $g_2^k = g_\psi^k + x_k$)

$$\begin{aligned} x_{k+1} &= \arg\min_{x \in \mathbb{R}^n} \varphi(x) + \|x\|^2/2 - \langle g_\psi^k + x_k + \gamma(x_k - x_{k-1}), x\rangle \\ &= \arg\min_{x \in \mathbb{R}^n} \varphi(x) + \frac{1}{2}\left\|x - (x_k + g_\psi^k + \gamma(x_k - x_{k-1}))\right\|^2 \\ &:= (I + \partial\varphi)^{-1}(x_k + g_\psi^k + \gamma(x_k - x_{k-1}))\,. \end{aligned}$$

As a result, this choice of parameter makes Algorithm 4 a linearized proximal method with inertia force. This allows us to rely on the analysis of [146, § 4.6] to establish the rate of convergence of this method applied to the DC program (3.1.1). To this end, we denote by $r(x)$ the following residuum

$$r(x) := x - (I + \partial\varphi)^{-1}(x + g_\psi)\,, \quad \text{with } g_\psi \in \partial\psi(x).$$

Notice that if $r(x_k) = 0$, then x_k solves $\min_{x\in\mathbb{R}^n} \varphi(x) + \frac{1}{2}\left\|x - (x_k + g_\psi^k)\right\|^2$. Its optimality condition yields $g_\psi^k \in \partial\varphi(x_k)$, showing that x_k is a critical point for problem (3.1.1) (which reads for this particular setting as $\min_{x\in\mathbb{R}^n} \varphi(x)-\psi(x)$). The following result shows that the rate of convergence of both sequences $\{\|r(x_k)\|^2\}$ and $\{\|x_{k+1} - x_k\|^2\}$ is $\mathcal{O}(\frac{1}{k})$.

Theorem 3.4.4. *([51, Theorem 2]). Suppose that the level set $L_f(x_0) := \{x \in \mathbb{R}^n : f(x) \leq f(x_0)\}$ is bounded, $\gamma \in [0, 1/2)$ and let $\bar{f} := \min_{L_f(x_0)} f(x)$ and $k > 0$. Then*

$$(a) \qquad \min_{i\in\{0,\ldots,k\}} \|x_{i+1} - x_i\|^2 \leq \left(\frac{2}{1-2\gamma}\right)\frac{f(x_0) - \bar{f}}{k+1}$$

and

$$(b) \qquad \min_{i\in\{0,\ldots,k\}} \|r(x_i)\|^2 \leq \left(\frac{16}{1-2\gamma}\right)\frac{f(x_0) - \bar{f}}{k+1}.$$

3.5 Application of interest: nonconvex image denoising

Image reconstruction techniques have become important tools in computer vision systems and many other applications that require sharp images obtained from noisy/corrupted ones. The convex total variation (TV) formulations have proven to provide a good mathematical basis for several basic operations in image reconstruction [36]. In order to present such formulations, let $b \in \mathbb{R}^n$ be the vectorization of a corrupted $n_1 \times n_2$ *grayscale* image B (in this case, $n = n_1 \cdot n_2$). A TV formulation, with penalizing function $\phi : \mathbb{R}_+ \to \mathbb{R}$, consists in solving the following minimization problem

$$\min_{x\in\mathbb{R}^n} \frac{\mu}{2}\|x - b\|^2 + TV_\phi(x), \quad \text{with} \quad TV_\phi(x) := \sum_{i=1}^{n} \phi(\|(\nabla x)_i\|), \tag{3.5.1}$$

where $\mu > 0$ is a fidelity parameter and $(\nabla x)_i \in \mathbb{R}^2$ denotes the discretization of the gradient of image x at pixel i, that is, $(\nabla x)_i$ represents finite difference approximations of first-order horizontal and vertical partial derivatives. In a matrix representation $X \in \mathbb{R}^{n_1\times n_2}$ of $x \in \mathbb{R}^n$ we have that

$$(\nabla x)_i := \begin{pmatrix} X_{l+1,j} - X_{l,j} \\ X_{l,j+1} - X_{l,j} \end{pmatrix}, \quad \text{with } i^{th} \text{ the coordinate of } x \text{ where the pixel } X_{l,j} \text{ is stored.}$$

Thus $\|(\nabla x)_i\| = \sqrt{(X_{l+1,j} - X_{l,j})^2 + (X_{l,j+1} - X_{l,j})^2}$.

If the penalizing function is chosen to be $\phi(r) = r$, then problem (3.5.1) consists in a convex nonsmooth optimization problem that can be efficiently solved by several specialized algorithms such the ones proposed in [13, 20, 41]. This is the main benefit of using a convex formulation for image denoising. Nevertheless, nonconvex regularizations have remarkable advantages over convex ones for restoring images, in particular, high-quality piecewise constant images with neat edges [144]. In order to preserve edges in the restoration process, some authors [111, 126] employ a nonconvex penalizing function ϕ to induce sparse image gradients $(\nabla x)_i$. This makes (3.5.1) a nonconvex and nonsmooth problem, that has been recently dealt with by local (strongly) convex approximations in [111].

In what follows we show that for a wide class of penalizing functions $\phi : \mathbb{R}_+ \to \mathbb{R}$, problem (3.5.1) is indeed a DC programming problem with available DC decompositions.

3.5.1 DC decomposition of nonconvex denoising models

Assume that $\phi : \mathbb{R}_+ \to \mathbb{R}$ is a concave and non-decreasing function. As a result, its right derivative is well defined for all $r \geq 0$, that is, the limit $\phi'_+(r) := \lim_{h\downarrow 0} \frac{\phi(r+h)-\phi(r)}{h}$ exists for all $r \geq 0$. Our goal is to prove that under these assumptions, the composite function $TV_\phi(x) = \sum_{i=1}^n \phi(\|(\nabla x)_i\|)$ can be written as a difference of two convex functions. To this end, we will need the following useful result, whose proof can be obtained by combining some developments presented in [196, § 4].

Lemma 3.5.1. *([51, Lemma 3]). Let $c : \mathbb{R}^n \to \mathbb{R}_+$ be a convex function. If $\phi : \mathbb{R}_+ \to \mathbb{R}$ is a concave and non-decreasing function such that $\phi'_+(0) < \infty$, then $\tau\, c(x) - \phi(c(x))$ is convex for all $\tau \geq \phi'_+(0)$.*

Proposition 3.5.2. *([51, Proposition 2]). Let $\phi : \mathbb{R}_+ \to \mathbb{R}$ be as in Lemma 3.5.1, and the TV function given by $TV(x) := \sum_{i=1}^n \|(\nabla x)_i\|$. Then $\tau\, TV(x) - TV_\phi(x)$ is a convex function for all $\tau \geq \phi'_+(0) \geq 0$.*

Since the requirements on the penalizing function ϕ are mild, Proposition 3.5.2 is quite general. For instance, all the functions ϕ considered in [111] and reported in Table 3.2 satisfy the assumptions of Proposition 3.5.2 with $\tau \geq 1$. (This ensures that function f_2 in Example 3.2.2 is indeed convex.)

	$\phi_{\log}$	$\phi_{\texttt{rat}}$	$\phi_{\texttt{atan}}$	$\phi_{\exp}$
$\phi_a(r)$	$\frac{\log(1+ar)}{a}$	$\frac{r}{1+ar/2}$	$\frac{\texttt{atan}((1+ar)/\sqrt{3})-\pi/6}{a\sqrt{3}/2}$	$\frac{1-\exp(-ar)}{a}$
$\phi'_a(r)$	$\frac{1}{1+ar}$	$\frac{1}{(1+ar/2)^2}$	$\frac{1}{1+ar+a^2r^2}$	$\frac{1}{\exp(ar)}$

Table 3.2: Some examples of concave, differentiable and non-decreasing penalizing functions $\phi_a : \mathbb{R}_+ \to \mathbb{R}_+$ parameterized by $a > 0$.

3.5.2 Specific DC models

We now focus on subproblem (3.2.2) resulting from the considered nonconvex image denoising model. Without loss of generality, we assume in the remaining of this paper that the penalizing function $\phi : \mathbb{R}_+ \to \mathbb{R}$ satisfies $\phi'_+(0) = 1$ (this is the case of the functions presented in Table 3.2). In this setting, the DC function reads as

$$f(x) = \frac{\mu}{2}\|x-b\|^2 + TV_\phi(x) = \underbrace{\frac{\mu+\rho}{2}\|x-b\|^2 + TV(x)}_{f_1(x)} - \underbrace{\left[\frac{\rho}{2}\|x-b\|^2 + TV(x) - TV_\phi(x)\right]}_{f_2(x)} \tag{3.5.2}$$

In the above definition, $\rho > 0$ is an arbitrarily chosen parameter to satisfy Assumption A1. In our numerical experiments we set $\rho = 1$. Since f is nonnegative and centered around b, f is coercive and thus has bounded-level sets, satisfying thus the assumption of Theorem 3.4.3.

With this formulation, subproblem (3.2.2) reads as

$$\min_{x\in\mathbb{R}^n} \frac{\mu+\rho}{2}\left\|x-\zeta^k\right\|^2 + TV(x), \quad \text{with } \zeta^k := b + (g_2^k + \gamma(x_k - x_{k-1})/(\mu+\rho)\,, \tag{3.5.3}$$

i.e., a convex denoising problem with corrupted image b perturbed by $(g_2^k + \gamma(x_k - x_{k-1})/(\mu+\rho)$. As already mentioned this subproblem can be efficiently solved by several specialized methods [13, 20, 41].

3.6 Numerical results

In this section, we consider the nonconvex image denoising model (3.5.1) with penalizing function $\phi_{\texttt{atan}}$ given by $\phi(r) := \frac{\texttt{atan}((1+ar)/\sqrt{3})-\pi/6}{a\sqrt{3}/2}$ with $a = 4$. Given that similar results can be achieved with the remaining functions of Table 3.2, adequately tuned, we only provide detailed results for this penalizing function. Since $\phi'_+(0) = 1$, Proposition 3.5.2 ensures that the objective function of (3.5.1) has the DC decomposition $f_1 - f_2$, with f_1 and f_2 given in (3.5.2).

In our numerical experiments we set $\rho = 1$ in (3.5.2) and $\lambda = 0$ in Algorithm 4. As a result, condition (3.3.1) is equivalent (for $\lambda = 0$) to define the next iterate x_{k+1} as a solution of the convex image denoising problem (3.5.3) with corrupted image perturbed by an inertial force. This task is accomplished at every iteration of our algorithm by employing the convex nonsmooth denoising method known as FISTA (Fast Iterative Shrinkage/Thresholding Algorithm) [20]. We have also tested our algorithm with $\lambda > 0$

employing the bundle-method's idea described in Algorithm 5. However, for this class of problems, the resulting DC bundle algorithm was not competitive with its exact counterpart employing FISTA. Hence, we do not report results on inexact variants of Algorithm 4. In what follows we examine the numerical performance of the following solvers, all of them coded in MATLAB version R2015b:

- `DCA` - Algorithm 4 with $\gamma = \lambda = 0$. The trial point x_{k+1} is defined by solving the convex subproblem (3.5.3) with a MATLAB implementation of FISTA[1].
- `InDCA` - The same as `DCA`, but with inertial factor $\gamma = 0.499$ instead.
- $\texttt{InDCA}_{\gamma_k}$ - The same as `DCA`, but varying the inertial factor $\gamma > 0$ along the iterative process. We initialize the solver with $\gamma = 2.5$ and set

$$\gamma \leftarrow \max\{0.499,\, \gamma/2\} \quad \text{whenever} \quad f(x_{k+1}) + \frac{1-\gamma}{2}\left\|x_{k+1} - x_k\right\|^2 > f(x_k) + \frac{\gamma}{2}\left\|x_k - x_{k-1}\right\|^2 .$$

  The reasoning behind this rule is given by Lemma 3.4.2, which ensures $f(x_{k+1}) + \frac{1-\gamma}{2}\left\|x_{k+1} - x_k\right\|^2 \leq f(x_k) + \frac{\gamma}{2}\left\|x_k - x_{k-1}\right\|^2$ for all $\gamma \in [0, 1/2)$ (because we set $\rho = 1$). However, due to the nature of function f_2, Assumption A1 may hold for a larger ρ (this is why start with $\gamma = 2.5$ instead of $\gamma < 1/2$). If the inequality in the above rule holds and $\gamma \geq 1/2$, then γ is found to be too large to ensure convergence of the algorithm. We thus must reduce γ until becoming lower than the threshold $\rho/2 = 1/2$.
- `iPiano` - *Inertial proximal algorithm for nonconvex optimization.* This is an implementation of Algorithm 2 given in [146], with the inertial parameter therein fixed to[2] 0.8, and constants L, α given by 100 and 0.003, respectively. This choice of parameters was made upon some tuning. This solver requires function $-f_2$ to be differentiable and Lipschitz continuous, which is not the case of f_2 given in (3.5.2). Therefore, for `iPiano` only, we replaced f_2 with the convex and differentiable function $f_2^{\texttt{smooth}}(x) := \frac{\mu}{2}\left\|x - b\right\|^2 + \sum_{i=1}^n [s(\|(\nabla x)_i\|) - \phi(s(\|(\nabla x)_i\|))]$, where $s(t) = \sqrt{t^2 + c^2} - c$ is the pseudo-Huber function with parameter $c = 10^{-3}$. Once an oracle provides the gradient g^k of $\sum_{i=1}^n [s(\|(\nabla x)_i\|) - \phi(s(\|(\nabla x)_i\|))]$ at $x = x_k$, the next iterate is computed in a closed form: $x_{k+1} := [\alpha\mu b + (x_k - g^k + \gamma(x_k - x_{k-1}))]/(1 + \alpha\mu)$. Therefore, when compared to the DC solvers above, `iPiano` possesses a much lower computational burden per iteration. Although we considered $f_2^{\texttt{smooth}}$ along the optimization process of `iPiano`, the function values provided in the tables below correspond to the function $f = f_1 - f_2$ (without smoothing).

All these solvers employ the same black-box for f_2 and the same stopping-test: the iterative process terminates when the inequality

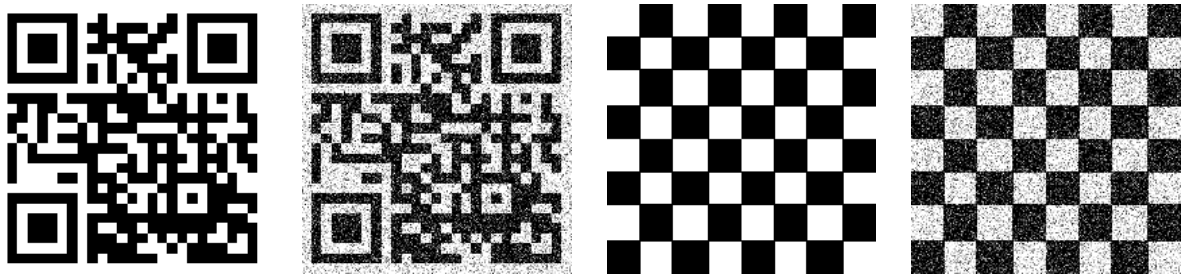
$$\max\{\left\|x_{k+1} - x_k\right\|,\, \gamma\left\|x_k - x_{k-1}\right\|\} \leq 5 \times 10^{-4}\,(1 + \left\|x_{k-1}\right\|) \quad \text{is satisfied.}$$

We set the maximum number of iterations of the DC solvers to 100. Since `iPiano` has a low computational burden per iteration, its maximum number of iterations was fixed to 1000.

The numerical performances of these four nonconvex solvers (with three of them exploiting the DC decomposition (3.5.2) of (3.5.1)) are assessed on two piecewise-constant images corrupted by a Gaussian noise with mean 0 and variance 0.1. Figures 3.3(a) and 3.3(c) present the original (non-corrupted) images whereas Figures 3.3(b) and 3.3(d) show the corrupted ones. Each image has dimension 200×200, which yields large-scale nonsmooth DC optimization problems of dimension $n = 40\,000$. Numerical experiments were performed on a computer with Intel(R) Core(TM), i3-3110M CPU 2.40, 4G (RAM), under Windows 10, 64Bits.

[1] Available at `https://web.iem.technion.ac.il/images/user-files/becka/papers/tv_fista.zip`

[2] For this algorithm, the inertial parameter needs to be less than one.



(a) QR code Original (b) QR code Corrupted (c) Checkerboard Original (d) Checkerboard Corrupted

Figure 3.3: Piecewise constant images. The corrupted images were obtained by the original ones by adding a Gaussian noise. All images have dimension 200×200.

In order to check the quality of the restored images we employ two well-known measures in the community of computational vision: the *peak signal-to-noise ratio* PSNR and the *structural similarity* SSIM. Detailed descriptions of these measures can be found in [78, 206]. For our purposes, it is enough to keep in mind that the larger the PSNR is the better is the restoration. The same indication can be yielded when the SSIM is closer to one. Nevertheless, the focus here is on CPU time and function values provided by considered solvers: we aim at illustrating the performances of the new proposal, rather than investigating technical matters of Digital Images/Computational Vision.

We start by presenting in Table 3.3 the results obtained by applying the four considered solvers to the nonconvex image denoising model (3.5.1) issued by the corrupted QR-code image of Figure 3.3(b). In our experiments, the fidelity parameter μ ranges from 0.55 to 1.25.

The results show that the inertial DC algorithms provide lower function values in less CPU time/subgradient evaluations. Concerning these features, the performance of `InDCA` was better than the one of `DCA`. Moreover, solver $\texttt{InDCA}_{\gamma_k}$ (that has a larger inertial factor) significantly outperformed `DCA` in this application. For instance, for the case $\mu = 1.15$ solver $\texttt{InDCA}_{\gamma_k}$ found a critical point with function value around 1.18% lower than the function value provided by `DCA`. Furthermore, in this case, solver $\texttt{InDCA}_{\gamma_k}$ was almost 50% faster than `DCA`.

In Figure 3.4 we present the restored images obtained by the solvers applied to the instance with $\mu = 0.85$ (the one with highest values of PSNR and SSIM). In addition to the nonconvex models, we present in

(a) Convex model
$PSNR = 19.453$
$SSIM = 0.813$
$f(x_k) = 10334.721$

(b) `iPiano`
$PSNR = 21.355$
$SSIM = 0.888$
$f(x_k) = 9989.072$

(c) `DCA`
$PSNR = 21.687$
$SSIM = 0.893$
$f(x_k) = 9762.382$

(d) `InDCA`
$PSNR = 21.731$
$SSIM = 0.894$
$f(x_k) = 9755.709$

(e) $\texttt{InDCA}_{\gamma_k}$
$PSNR = 21.835$
$SSIM = 0.902$
$f(x_k) = 9686.490$

Figure 3.4: Restored QR-code images. Convex model with fidelity parameter $\mu = 3.75$ and nonconvex model with $\mu = 0.85$. For comparison reasons, the final value $f(x_k)$ in (a) was computed with $\mu = 0.85$.

Figure 3.4(a) the image restored by employing a convex model (resulting from setting $\phi(r) := r$ in (3.5.1), i.e., $TV(x)$ instead of $TV_\phi(x)$). The convex denoising model was solved by algorithm FISTA [20]. Among all the considered values of the fidelity parameter μ in (3.5.1), the best results were

Solver	μ	CPU(s)	# g_2	$f(x_{\texttt{best}})$	PSNR	SSIM
`iPiano`	0.55	469	1000	9322.314	21.687	0.882
`DCA`	0.55	122	58	9084.983	21.931	0.882
`InDCA`	0.55	111	53	9084.814	21.855	0.881
`InDCA`$_{\gamma_k}$	0.55	62	26	9045.975	19.253	0.839
`iPiano`	0.65	477	1000	9543.678	21.719	0.890
`DCA`	0.65	223	100	9302.109	22.033	0.889
`InDCA`	0.65	223	100	9300.447	21.997	0.888
`InDCA`$_{\gamma_k}$	0.65	140	60	9263.575	21.332	0.874
`iPiano`	0.75	473	1000	9763.515	21.591	0.891
`DCA`	0.75	234	100	9528.582	21.951	0.892
`InDCA`	0.75	189	80	9526.251	21.935	0.892
`InDCA`$_{\gamma_k}$	0.75	242	100	9476.832	21.817	0.897
`iPiano`	0.85	479	1000	9989.072	21.355	0.888
`DCA`	0.85	213	88	9762.382	21.687	0.893
`InDCA`	0.85	190	79	9755.709	21.731	0.894
`InDCA`$_{\gamma_k}$	0.85	109	45	9686.490	21.835	0.902
`iPiano`	0.95	472	1000	10209.231	21.107	0.884
`DCA`	0.95	174	72	9995.409	21.323	0.891
`InDCA`	0.95	137	57	9993.745	21.390	0.890
`InDCA`$_{\gamma_k}$	0.95	95	39	9900.572	21.505	0.876
`iPiano`	1.05	470	1000	10434.161	20.810	0.876
`DCA`	1.05	192	80	10221.634	21.103	0.888
`InDCA`	1.05	134	56	10220.961	21.055	0.887
`InDCA`$_{\gamma_k}$	1.05	84	35	10118.874	21.453	0.879
`iPiano`	1.15	472	1000	10672.028	20.545	0.864
`DCA`	1.15	163	67	10461.936	20.738	0.879
`InDCA`	1.15	162	67	10452.538	20.785	0.880
`InDCA`$_{\gamma_k}$	1.15	82	34	10338.301	21.263	0.879
`iPiano`	1.25	487	1000	10909.249	20.226	0.856
`DCA`	1.25	162	68	10706.295	20.433	0.872
`InDCA`	1.25	158	66	10694.461	20.479	0.875
`InDCA`$_{\gamma_k}$	1.25	102	42	10547.789	20.932	0.858

Table 3.3: Restoration of the corrupted QR-code image. CPU times are given in seconds. The notation # g_2 stands for the number of subgradient evaluations of function f_2 (or $f_2^{\texttt{smooth}}$). This coincides with the number of iterations performed by the algorithms. The maximum number of subgradient evaluations was set to 1000 for solver `iPiano` and to 100 for the other solvers.

obtained with $\mu = 3.75$ for the convex model. The quality-measure values in Figure 3.4(a) indicates that convex models are not effective to preserve edges in the restoration process of piecewise-constant images, corroborating thus with the conclusion drawn in [144]. Moreover, the quality of the restored image of Figure 3.4(a) is visibly worse than the one restored by solver `InDCA`$_{\gamma_k}$. In fact, Figure 3.4(e) contains less noise than the images restored by solvers `iPiano`, `DCA` and `InDCA`.

In what follows we examine the corrupted checkerboard image of Figure 3.3(d). Table 3.4 contains some results obtained by applying the four considered solvers to this image by varying the fidelity parameter μ from 0.8 to 1.6.

Once again, the inertial DC solver `InDCA`$_{\gamma_k}$ provided better results than the other considered solvers. The inertial solvers required fewer iterations than `DCA` to terminate with a critical point of better quality (except for the instance $\mu = 0.9$, at which solver `InDCA` performed more iterations than `DCA`).

In Figure 3.5 we present the restored images obtained by the solvers applied to the instance with $\mu = 1$. Once again, the convex denoising model was solved by algorithm FISTA.

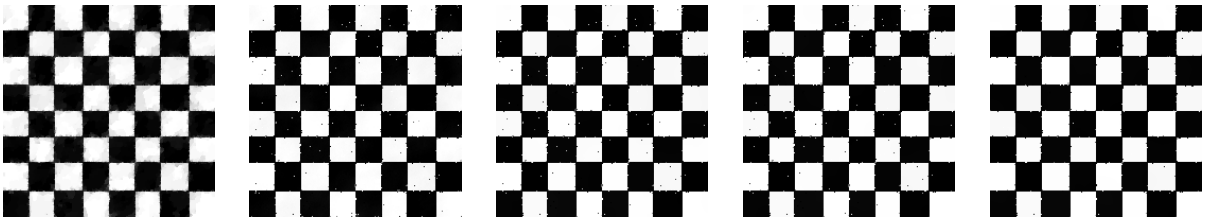
3.6.1 Assessing numerical performance on several instances

In order to assess the numerical performances of the considered DC solvers we examine 72 instances of the nonconvex image denoising model (3.5.1), obtained by varying μ as in Table 3.4 and by considering four different penalizing functions ϕ as in Table 3.2, and the two corrupted images of Figure 3.3.

We present the performance profiles [61] of `DCA`, `InDCA` and `InDCA`$_{\gamma_k}$ on these 72 instances. As an

Solver	μ	CPU(s)	# g_2	$f(x_{\texttt{best}})$	PSNR	SSIM
`iPiano`	0.80	474	1000	9026.884	23.879	0.820
`DCA`	0.80	105	56	8778.145	24.049	0.826
`InDCA`	0.80	81	44	8779.285	24.160	0.826
`InDCA`$_{\gamma_k}$	0.80	110	58	8716.393	24.239	0.772
`iPiano`	0.90	479	1000	9251.406	23.481	0.812
`DCA`	0.90	81	44	9014.267	23.670	0.822
`InDCA`	0.90	86	47	9006.346	23.702	0.824
`InDCA`$_{\gamma_k}$	0.90	60	32	8921.761	23.862	0.734
`iPiano`	1.00	474	1000	9475.528	23.166	0.801
`DCA`	1.00	112	61	9237.498	23.337	0.811
`InDCA`	1.00	106	57	9232.004	23.365	0.812
`InDCA`$_{\gamma_k}$	1.00	56	30	9138.142	24.230	0.849
`iPiano`	1.10	470	1000	9708.161	22.730	0.786
`DCA`	1.10	114	61	9481.109	22.952	0.799
`InDCA`	1.10	105	56	9474.022	22.968	0.801
`InDCA`$_{\gamma_k}$	1.10	52	28	9326.864	24.059	0.773
`iPiano`	1.20	472	1000	9949.721	22.205	0.761
`DCA`	1.20	109	58	9727.282	22.451	0.782
`InDCA`	1.20	92	49	9719.249	22.511	0.783
`InDCA`$_{\gamma_k}$	1.20	63	34	9536.481	23.969	0.803
`iPiano`	1.30	476	1000	10187.130	21.754	0.739
`DCA`	1.30	140	73	9968.101	21.992	0.765
`InDCA`	1.30	134	70	9959.879	22.045	0.768
`InDCA`$_{\gamma_k}$	1.30	77	42	9757.747	23.434	0.794
`iPiano`	1.40	475	1000	10424.813	21.317	0.714
`DCA`	1.40	149	75	10215.140	21.553	0.742
`InDCA`	1.40	116	59	10208.485	21.593	0.744
`InDCA`$_{\gamma_k}$	1.40	55	29	10035.034	22.944	0.822
`iPiano`	1.50	476	1000	10673.945	20.855	0.684
`DCA`	1.50	165	85	10473.202	21.082	0.713
`InDCA`	1.50	127	64	10464.077	21.091	0.716
`InDCA`$_{\gamma_k}$	1.50	61	31	10277.361	22.424	0.808
`iPiano`	1.60	476	1000	10921.842	20.390	0.661
`DCA`	1.60	155	80	10731.337	20.626	0.683
`InDCA`	1.60	136	70	10724.553	20.635	0.687
`InDCA`$_{\gamma_k}$	1.60	54	28	10524.680	21.858	0.786

Table 3.4: Restoration of the corrupted checkerboard image. The notation # g_2 stands for the number of subgradient evaluations of function f_2 (or $f_2^{\texttt{smooth}}$). This coincides with the number of iterations performed by the algorithms. The maximum number of subgradient evaluations was set to 1000 for solver `iPiano` and to 100 for the other solvers.



(a) Convex model
$PSNR = 22.101$
$SSIM = 0.725$
$f(x_k)$=9513.226

(b) `iPiano`
$PSNR = 23.166$
$SSIM = 0.801$
$f(x_k) = 9475.528$

(c) `DCA`
$PSNR = 23.337$
$SSIM = 0.811$
$f(x_k) = 9237.498$

(d) `InDCA`
$PSNR = 23.365$
$SSIM = 0.812$
$f(x_k) = 9232.004$

(e) `InDCA`$_{\gamma_k}$
$PSNR = 24.230$
$SSIM = 0.849$
$f(x_k) = 9138.142$

Figure 3.5: Restored checkerboard images. Convex model with fidelity parameter $\mu = 3.4$ and nonconvex model with $\mu = 1$. For comparison reasons, the final value $f(x_k)$ in (a) was computed with $\mu = 1$.

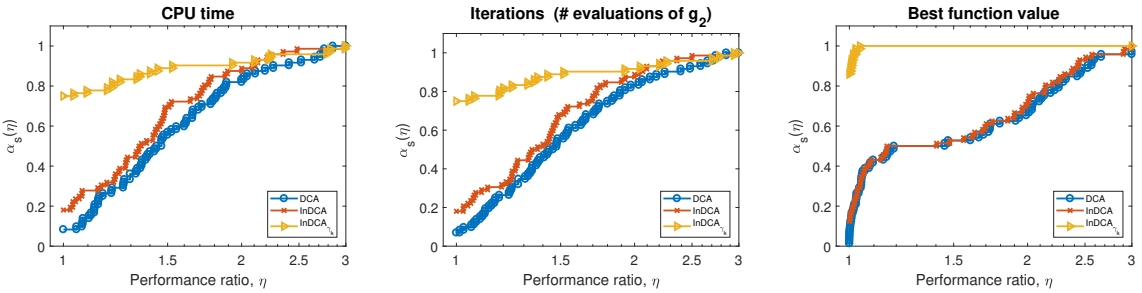
example, let the criterion of analysis be CPU time. For each solver, we plot the proportion of instances that is solved within a factor η of the time required by the best algorithm. More specifically, denoting by $t_s(i)$ the time spent by solver s to solve instance i and by $t^*(i)$ the best time for the same instance among all the solvers, the proportion of instances solved by s within a factor η is

$$\alpha_s(\eta) := \frac{\text{number of instances } i \text{ such that } t_s(i) \leq \eta\, t^*(i)}{\text{total number of instances}}.$$

Therefore, the value $\alpha_s(1)$ gives the probability of the solver s to be the best by a given criterion. Furthermore, unless $t_s(i) = \infty$ (which means that solver s failed to solve instance i), it follows that $\lim_{\eta\to\infty} \alpha_s(\eta) = 1$. Thus, the higher is the line, the better is the solver (by this criterion).

Figure 3.6.1(a) presents the performance profile of the solvers with respect to CPU time. Solver $\texttt{InDCA}_{\gamma_k}$ was the fastest one in 74% of the considered instances, followed by `InDCA` (18%). A similar conclusion can be drawn concerning the number of subgradient evaluations of f_2 (that coincides with the number of iterations): Figure 3.6.1(b) shows that $\texttt{InDCA}_{\gamma_k}$ was the solver that required less subgradient evaluations in 75% of the instances.

We recall that the optimal values of the considered instances of problem (3.5.1) are unknown. To assess the quality of the solutions computed by the solvers we proceed as follows. Let $\bar{f}_i^s$ be the function value of instance i computed by solver s, and let $\bar{f}_i^{best} := \min_s \bar{f}_i^s$ be the best function value computed by the three solvers. In Figure 3.6.1(c) we plot the performance profile of the solvers with respect to the criterion[3] $\frac{\bar{f}_i^s - f_i^{best} + 1}{f_i^{best} + 1}$. Solver $\texttt{InDCA}_{\gamma_k}$ computed the best function value in 86% of the cases, followed by `InDCA` that was the most effective one in 12.5% of the instances.



(a) CPU time (b) Subgradient evaluations of f_2 (c) Critical points' quality

Figure 3.6: Performance profile over 72 instances of the nonconvex image denoising model (3.5.1).

These results show that furnishing the DC algorithm with an inertial force pays off: for the considered application we observed that the quality of the computed critical points improves whereas the CPU time decreases. Finally, we comment that the total CPU time to solve these 72 instances by `DCA` was 137 minutes, by `InDCA` 127 minutes, and by $\texttt{InDCA}_{\gamma_k}$ 108 minutes. The latter provided a CPU time reduction of around 21% concerning DCA.

3.7 Concluding remarks

With the purpose of computing critical points of better quality in (unconstrained or convex-constrained) DC programs we have equipped the classical DC algorithm with an inertial force. Convergence analysis and rate of convergence of the new proposal have been established. Moreover, we have investigated less demanding procedures to compute trial points and have shown that the given algorithmic pattern covers

[3] We added "1" to prevent this measure being zero.

and extends some well-known optimization methods found in the literature, such as the DCA, proximal linearized method, and DC bundle methods.

The numerical performance of two variants of the given algorithmic pattern was assessed on nonconvex and nonsmooth image denoising models yielding optimization problems of dimension 40 000. In this application, every iteration of our inertial DC algorithm amounts to solving a convex image denoising model with corrupted image perturbed by an inertial force. As presented in the numerical section, such a perturbed model can be efficiently solved by specialized approaches such as the FISTA algorithm. At least for this application, our numerical experiments indicate that the proposed algorithm outperforms the classic one in terms of CPU time, number of subgradient evaluations (of the second-component function) and, mainly, in terms of quality of the computed critical points.

Chapter 4

Optimization techniques for the Brazilian natural gas network planning problem

This chapter presents the following work that models uncertainties in the long-term design and operation planning problem of the Brazilian natural gas network:

> S.V.B. Bruno, L.A.M. Moraes and W. de Oliveira
> *Optimization techniques for the Brazilian natural gas network planning problem.*
> **Energy Systems** *(ENSY), 2017, volume 8, issue 1, pp. 81–101.*

The paper is one of the products of the collaboration between IMPA and PETROBRAS, the Brazilian oil and gas company. The considered problem is modeled as a two-stage stochastic linear program and solved by combining decomposition, a bundle method algorithm, and scenario reduction techniques. These strategies were implemented in the software `MONGE` that will assist, in 2019, the renegotiation of the 20-year gas supply contract between Brazil and Bolivia.

4.1 Introduction

Expansion and operation planning of the existing supply chain is a key activity in the natural gas market. By expansion, one can consider physical expansion of pipelines, construction of new ones, and negotiation of prices and volumes for supply and demand contracts – new or existent ones. There is also a need for an integration between expansion and operation plans, due to the quest for higher profitability. Thus, existent and new infrastructure must operate in an optimal way, considering the following options for natural gas use:

- *acquisition* – production or purchase of imported gas or liquefied natural gas (LNG) loads;
- *consumption* – local distribution companies demand, internal consumption, and thermoelectric power plants fueling.

Figure 4.1 shows an example of a natural gas supply chain that starts with exploration and production activities that, together with imports, correspond to gas *acquisition*. Gas is shipped to delivery nodes that represent *consumption*. Activities covered by the model presented in this work are marked with a rectangle.

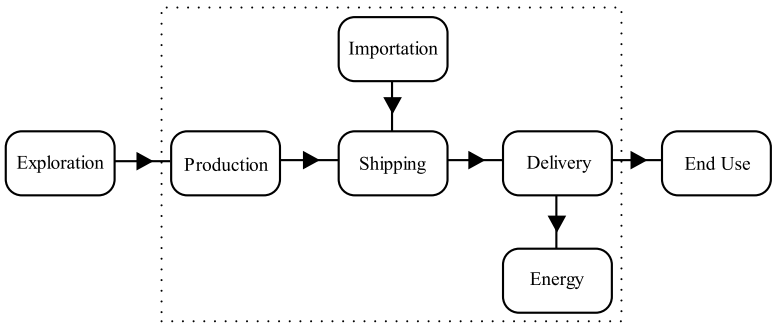


Figure 4.1: Activities of natural gas supply chain

Investments in new natural gas sources usually take up to ten years until production is initiated reliably and safely and involve large amounts of money. Thereby, long-term planning of a natural gas supply chain, taking into account infrastructure development and long-term operation of the system, is of great importance for the profitability of the natural gas industry, [124].

4.1.1 Natural gas industry: the Brazilian case

In Brazil, natural gas is expected to have a more relevant role in the energy market over the coming decades. Most of the Brazilian natural gas is associated to oil production and this associated volume will increase as soon as more pre-salt oil fields come into operation. In 2012 natural gas average supply was about 2.65 BCF/day, of which 1.39 BCF/day is related to domestic production, 0.95 BCF/day were imported from Bolivia and the remaining 0.31 BCF/day refers to LNG loads, specially from Nigeria and Trinidad and Tobago. These numbers correspond to a 22% increase, if compared to 2011 numbers.

It is expected that by 2019 Brazilian natural gas consumption will be about 6 BCF/day, of which 30% will be related to industrial sector and about 9.5% will be used for energy generation. In that same year the Brazil/Bolivia 20-year supply contract must be renegotiated.

Natural gas has several uses: in residences, in industries, in oil fields for re-injection purposes, as fuel for refineries and rigs, for vehicles, and for energy generation. Some of these demands can be deterministically forecast due to its low variability, such as residences, industries, and vehicles consumption. However, gas demand for energy generation purposes can be highly uncertain, as explained below.

4.1.1.1 Natural gas for power production: source of uncertainty for gas demand

The Brazilian power generating system is hydro dominated and characterized by large reservoirs with multi-year regulation capability, arranged in complex cascades over several river basins. According to the *Brazilian Electricity Regulatory Agency* (ANEEL), the hydroelectric plants were responsible for about 62% of the total energy generation in 2015[1]. Due to this hydro predominance, operation is driven by the rainfall events – occurrence and forecasting. Therefore, thermoelectric dispatch (and thus gas consumption) depends on the rainfall, which is uncertain.

As an example, if hydroelectric reservoirs are at low levels and a dry period (in terms of rainfall) is expected, then thermoelectric dispatch is high, in order to save water in the hydroelectric reservoirs. On the other hand, if reservoirs are at regular or high levels, and no dry period is being expected in the coming months, so thermoelectric dispatch tends to be small because there is no need to save water.

In Brazil, thermoelectric dispatch is centralized by an independent system operator (ONS) supported

[1] `http://www.aneel.gov.br/aplicacoes/capacidadebrasil/capacidadebrasil.cfm`

by the computational model NEWAVE, the official optimization program for operating and planning the Brazilian power system [130]. One of the outputs of this model is a set of scenarios of thermoelectric dispatch that can be used to represent the stochasticity of the gas demand for energy generation purposes.

4.1.1.2 Accounting the uncertainties and modeling of the problem

The stochastic process of gas demand in Brazil can be approximated by a set of scenarios extracted from the thermoelectric dispatches provided by the optimization model NEWAVE. This opens the way to address the Brazilian natural gas network planning problem by adopting the framework of *stochastic programming with recourse*, as defined in [28]. An approach with *recourse* is suitable for the case of interest because one can always buy natural gas from the international market (at higher prices) to supply a possible deficit of natural gas in the domestic market.

In this work we consider a case based on a real-life natural gas industry problem from PETROBRAS, the Brazilian oil and gas company. Such problem, which is of large-scale and difficult to solve, was modeled as a two-stage stochastic linear problem, meaning that some decisions – investments in infrastructure and new contracts – are of *here-and-now* type, i.e., they do not depend directly on the uncertain data. In each scenario the decision maker can *wait-and-see* the uncertainty revealed, and plan the network by taking into account existing infrastructures and new investments. In this (second) stage, natural gas volumes, pipelines use and thermoelectric plants operation must be decided.

By adopting a continuous framework where decisions variables related to investments can take fractional values, we study a compromise between uncertainty representation and solvability for the considered application: considering a large number of gas demand scenarios the resulting problem is solved by applying a decomposition technique akin to the L-Shaped method [186], but using a bundle method instead. An additional approach consists in selecting a much smaller but *representative* amount of scenarios to represent the gas demand uncertainties. The representative scenarios are selected by applying the optimal scenario reduction technique developed in [82].

4.1.2 Related works

The use of optimization models for natural gas supply chain has been studied in a deterministic fashion in [85]. The authors have also explored a related natural gas design problem in a multistage stochastic programming approach in [84].

Several authors have studied the use of stochastic programming methods in different planning levels – strategic, tactical or operational. In [176] one can find an extensive list of examples of decomposition models applied to energy systems optimization.

The work [179] presents a stochastic model for supply chain network design under uncertainty. The goal is to route the flow of products from a supplier to customers, and to define which processing centers should be built. The modeling results in a mixed-integer linear problem, which is solved via sample average approximation, [107]. In [180] a supply chain design under uncertainty is also studied. The authors apply sample average approximation and Lagrangian relaxation techniques to the resulting two-stage stochastic problem. Cutting-plane [103] and bundle methods [94] are used to solve the Lagrangian dual problem. The article [183] addresses a multistage capacity expansion planning problem using Dantzig-Wolfe Decomposition. In [102] an application to energy markets is studied, considering some preliminary decomposition strategies. An investment heuristic based on the stochastic dual dynamic programming approach for electricity markets was proposed in [143].

Another example of integrated design and operation problem is presented by [125], where a large-scale mixed-integer problem is solved by the so called nonconvex generalized Benders decomposition algorithm [145].

4.1.3 Contributions and organization

A contribution of the present work is the modeling of the Brazilian natural gas network planning problem by considering uncertainties in gas demand for power generation. By accounting for these uncertainties the proposed model is able to represent in a better manner real-world variables, leading to more efficient mathematical tools to assist decision making in this strategic supply chain problem of significant importance for the economical development of Brazil. As an additional contribution we highlight the empirical comparison of the benefits of decomposition techniques and optimal scenario reduction, as well as their effect over the considered natural gas network planning problem.

The paper is organized as follows. In Section 4.2 some characteristics of the considered problem are explained and the mathematical modeling is presented. A stochastic approach for the problem is proposed in Section 4.3. In Section 4.4, a decomposition technique and a bundle method are considered for solving practical instances. Section 4.5 addresses the optimal scenario reduction technique. Some concluding remarks are reported in Section 4.6. Numerical experiments for a 20-year horizon planning problem are given throughout Sections 4.3, 4.4 and 4.5.

4.2 The long-term planning problem

4.2.1 Problem statement

The Brazilian natural gas network planning problem is composed of:

- natural gas supply nodes – representing domestic gas production, imported gas, and LNG loads;
- demand nodes – representing local distribution companies and internal consumption (e.g. fertilizers factories and refineries);
- pipelines for the gas transportation;
- compressors to help with the gas transportation through pipelines;
- natural gas processing units – NGPU; and
- thermoelectric plants.

As we are dealing with a long-term (20-years horizon) planning problem, operational constraints are modeled with far less details than they would be in a short-term model, but some of them are still present in the model, such as pipeline maximum allowed flows and the use of a *heat rate* to convert natural gas into energy at thermoelectric plants.

The main goal of the model is to determine an optimal investment plan, including pipeline capacity expansions (or pipeline construction), volumes for new demand contracts (with, e.g., local distribution companies) and optimal operation of the whole network, i.e., gas flows through pipelines, gas volumes sent to demand nodes and absorbed in supply nodes, and thermoelectric plants operation.

In the modeled problem, a company buys natural gas at some input nodes (supply nodes) and must carry this gas to demand nodes or thermoelectric plants through its own pipelines network. For the sake of simplicity, we consider that no gas loss occurs during transportation. Each demand node expects to receive a certain amount of gas, varying from a minimal to a maximal value. If the optimal decision is to deliver less than the minimal natural gas demand at a node, company must pay for this unsatisfied amount of gas. The demand of thermoelectric plants are related to their dispatch levels, i.e., energy dispatch is converted into gas demand taking into account generator machines' efficiency, given by their nominal heat rate.

Investments may be made in certain periods within the planning horizon and are offered in projects. A project may consist of any combination of investment in: new pipelines or expansion of existing ones,

minimal and maximal volumes for supplying and demand contracts. As an example, a project may consist of a new pipeline to carry gas from a developed field to a new demand node, also included in the project, with minimal and maximal demand values.

The expansion and operation planning problem consists of deciding (i) which investment projects must be made, when and at which level, and (ii) given an optimal investment policy, volumes of gas to be bought at supply nodes, delivered at demand nodes (including thermoelectric plants), and volumes carried by pipelines.

In Figure 4.2 a schematic representation of the problem is given. Supply nodes, where natural gas can be bought by the company, are represented by boxes with capital letter S, while demand nodes, where natural gas is delivered, are represented by boxes with capital letter D. Gas can also be delivered to thermoelectric plants, represented by capital letter T. The company's internal network is composed of pipelines, compressors, and natural gas processing units (NGPU), represented by capital letters C and U, respectively.

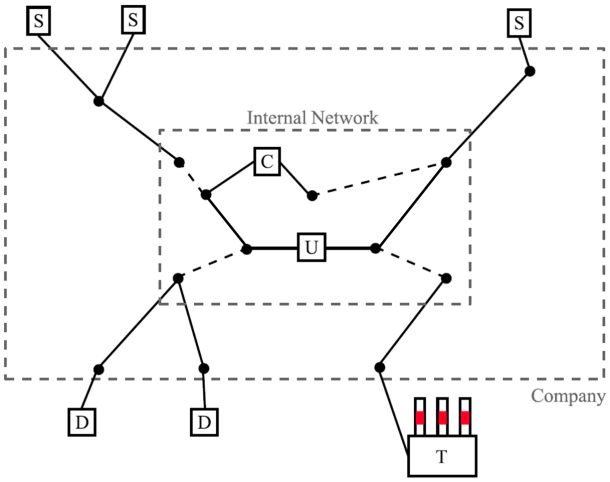


Figure 4.2: Natural gas network representation

4.2.2 Mathematical model

Let us first define two main sets: $\mathcal{N}$ and $\mathcal{P}$ stand for the set of nodes and pipelines, respectively. Each node $n \in \mathcal{N}$ can symbolize a gas input (resp. output) point, representing a supplier (resp. demand) or a concentration point, at which two or more pipeline segments converge to or diverge from. Two nodes can be connected by a pipeline segment, a compressor or a processing unit. Compressors and processing units are represented by sets $\mathcal{C}$ and $\mathcal{U}$, respectively. Each thermoelectric plant $\theta \in \Theta$ consists of a set of engines with different heat rates hr_θ. Supply and demand nodes are represented by sets $\mathcal{S} \subset \mathcal{N}$ and $\mathcal{D} \subset \mathcal{N}$. Naturally, the set of thermoelectric plants is contained in the set of demand nodes, i.e., $\Theta \subset \mathcal{D}$. Finally, let $\mathcal{I}$ denote the set of investment projects and $\mathcal{T}$ the set of periods. The decision variables of the considered problem are:

- $x_i^t \in [0,1]$ the percentage of investment i made up to period t;
- $z_i^t \in [0,1]$ amount of investment made at period t;
- $w_{ij}^t \geq 0$, the flow of gas from node i to node j at period t;
- $e_\theta^t \geq 0$, representing the amount of energy generated in plant θ, at period t; and

- $u_j^t \geq 0$, natural gas deficit in node j, period t.

The deterministic version of the considered natural gas network planning problem can now be stated and symbolized by

$$\min_{x,z,w,e,u} \sum_{t\in\mathcal{T}} \left\{ \sum_{i\in\mathcal{I}} ci_i \cdot z_i^t + \sum_{j\in\mathcal{S}} \left(cg_j^t \cdot \sum_{k\in\mathcal{N}} w_{jk}^t \right) + \right.$$
$$\left. + \sum_{j\in\mathcal{S}\cup D\setminus\Theta} cs_j \cdot u_j^t - \sum_{j\in\mathcal{D}\setminus\Theta} \left(rg_j^t \cdot \sum_{k\in\mathcal{N}} w_{kj}^t \right) - \sum_{\theta\in\Theta} re_\theta^t \cdot e_\theta^t \right\} \tag{4.2.1a}$$

$$\text{s.t.} \sum_{k\in\mathcal{N}} w_{jk}^t + u_j^t \geq v_j^t + \sum_{i\in\mathcal{I}} \left(x_i^t \cdot vi_{ij} \right), \quad \forall j \in \mathcal{S}, \forall t \in \mathcal{T} \tag{4.2.1b}$$
$$\sum_{k\in\mathcal{N}} w_{jk}^t \leq \upsilon_j^t + \sum_{i\in\mathcal{I}} \left(x_i^t \cdot vi_{ij} \right), \quad \forall j \in \mathcal{S}, \forall t \in \mathcal{T} \tag{4.2.1c}$$
$$\sum_{k\in\mathcal{N}} w_{kj}^t + u_j^t \geq v_j^t + \sum_{i\in\mathcal{I}} \left(x_i^t \cdot vi_{ij} \right), \quad \forall j \in \mathcal{D}\setminus\Theta, \forall t \in \mathcal{T} \tag{4.2.1d}$$
$$\sum_{k\in\mathcal{N}} w_{kj}^t \leq \upsilon_j^t + \sum_{i\in\mathcal{I}} \left(x_i^t \cdot vi_{ij} \right), \quad \forall j \in \mathcal{D}\setminus\Theta, \forall t \in \mathcal{T} \tag{4.2.1e}$$
$$w_{ij}^t \leq \phi_{ij}^t + \sum_{i\in\mathcal{I}} \left(x_i^t \cdot \phi i_{ij} \right) \quad \forall t \in \mathcal{T} \tag{4.2.1f}$$
$$hr_\theta \cdot e_\theta^t = \sum_{k\in\mathcal{N}} w_{k\theta}^t, \quad \forall \theta \in \Theta, \forall t \in \mathcal{T}, \tag{4.2.1g}$$
$$e_\theta^t \geq ed_\theta^t, \quad \forall \theta \in \Theta, t \in \mathcal{T}, \tag{4.2.1h}$$
$$z_i^t = x_i^t - x_i^{t-1} \geq 0, \quad \forall i \in \mathcal{I}, \forall t \in \mathcal{T}, \tag{4.2.1i}$$
$$w \in \mathcal{F}, e \in \Sigma, x \in \mathcal{X}. \tag{4.2.1j}$$

In this model, ci_i represents the total investment cost for project i and it is applied to the difference $x_i^t - x_i^{t-1}$ $(= z_i^t)$, which is equivalent to the amount of investment made at period t. Parameters cg_j and rg_j denote unitary acquisition cost and sale price for natural gas at node j, respectively. Additional cost term cs_j is included to penalize gas shortfall. Energy remuneration is represented by parameter re_θ, for thermoelectric plant θ. Thus, objective function (4.2.1a) consists of minimizing investment, gas acquisition and penalty costs minus demand and thermoelectric remuneration. In other words, solving the above problem is equivalent to maximizing the total profit of the company subject to operational and investment constraints.

Constraints (4.2.1b) and (4.2.1c) require minimum and maximum values of natural gas acquisition to be satisfied. Parameters v_j^t and vi_{ij} represent minimum amounts of natural gas to be bought – the former one represents original minimal value, and the later one represents the increase in minimal values due to investments. Constraints (4.2.1d) and (4.2.1e) represent the same requirements for demand nodes, *mutatis mutandis*. Pipeline capacities are represented in constraint (4.2.1f).

Energy generation is represented by constraints (4.2.1g) and (4.2.1h) – the former one shows the relation between natural gas consumption, $\sum w_{k\theta}^t$, and energy generation e_θ^t at thermoelectric plant θ, and the latter one ensures a minimal generation, representing energy dispatch. In constraint (4.2.1i), investments are forced to be non-decreasing, i.e., disinvestment is not allowed. Sets $\mathcal{F}$, Σ, and $\mathcal{X}$, in constraint (4.2.1j) represent, respectively, bounds on the flow of gas, bounds on energy generation of each plant θ, and bounds on the amount of investments for each period of the whole planning horizon.

The presented model is essentially a natural gas flow model, since thermoelectric dispatch is represented by equivalent natural gas demand. For our long term model we will assume an average operational heat rate for each unit of the considered thermoelectric power plants. In reality there is a heat rate curve

given by the load level of the unit, which must be accounted for in shorter time scales, such as real time operational decisions.

The model is general enough to represent onshore and offshore gas production, LNG supply and imports by the Brazil-Bolivia pipeline. Periods will have a duration ranging from one month up to a year.

4.2.3 Compact formulation

For convenience, model (4.2.1a)-(4.2.1j) is represented with the more compact notation

$$\begin{cases} \min\limits_{x,y} & \langle c, x\rangle + \langle q, y\rangle \\ \text{s.t.} & Tx + Wy \leq h \\ & x \in X,\, y \in Y\,, \end{cases} \tag{4.2.2}$$

where vector x keeps representing investment decisions, and planning decisions are symbolized by $y = (w \;\; e \;\; u)^\top$. Thus, the cost vector c is a rearrangement of vector ci, while q is defined by

$$q = ((cg - rg) \quad -re \quad cs)^\top.$$

Constraints (4.2.1b)-(4.2.1f) are represented in (4.2.2) by $Tx + Wy \leq h$ (note that constraint (4.2.1g) can easily be included in the latter matrix-vector formulation). Vector h is composed of parameters v, υ, ϕ, and ed. The polyhedral set X represents (4.2.1i) and the operational constraints $x \in \mathcal{X}$. The constraints $w \in \mathcal{F}$, $e \in \Sigma$ given in (4.2.1j) and $u \geq 0$ are represented in (4.2.2) by the compact form $y \in Y$.

For practical interests, problem (4.2.2) (and thus (4.2.1)) is too simplistic. In fact, some parameters involved in problem (4.2.2) may not be considered deterministic, such as the energy dispatch represented by parameter ed^t_θ whose reasons were presented in Section 4.1. Thereby, the deterministic model symbolized by problem (4.2.2) needs to be examined in a stochastic context.

4.3 Incorporating stochasticity into the problem

Concerning the Brazilian natural gas planning problem, the question we address in this section is the following one: Is there a significant gain by introducing stochasticity into the model represented by problem (4.2.1)?

In order to answer this question we consider energy dispatch scenarios to represent the uncertainty related to the natural gas demand ed^t_θ in (4.2.1h). One might also consider uncertainty in gas supplying, since the availability of some sources depend on exploratory success of natural gas and oil fields. We will however refrain from modeling this latter kind of uncertainty, since in the Brazilian case the gas supplying is much less uncertain than the natural gas demand.

As already mentioned in § 4.1.1.1, a set of N gas demand scenarios $\omega_1, \omega_2, \ldots, \omega_N$, with associated probability $\pi_i > 0$ for all $i = 1, \ldots, N$ is available by the computational program NEWAVE, described in [130]. Therefore, by adopting the framework of stochastic programming with recourse we can write the problem of interest in the following *deterministic equivalent* form, which is a *linear programming problem* (LP):

$$\begin{cases} \min\limits_{x,y_i} & \langle c, x\rangle + \sum_{i=1}^{N} \pi_i[\langle q, y_i\rangle] \\ \text{s.t.} & Tx + Wy_i \leq h(\omega_i) \\ & x \in X,\, y_i \in Y \text{ for all } i = 1, \ldots, N. \end{cases} \tag{4.3.1}$$

In order to evaluate the dependency of this problem on the gas demand scenarios, we solve an instance of the considered planning problem for several samples of scenarios. In what follows we describe the problem's instance.

4.3.1 Test problem

The test problem is a realistic instance of the Brazilian network planning problem, composed of 65 supply nodes, 305 demand nodes, 33 thermoelectric plants and 68 pipeline segments. The planning horizon is 20 years, split into 76 planning periods – each time period can represent either a month or a whole year, depending on the proximity of this period. The thermoelectric plants are centrally dispatched by the Brazilian independent system operator (ONS) and they will generate energy whenever the spot price is higher than their marginal operation cost. Nine investments projects are offered: four of them are related to the expansion of pipeline capacities, three represent new supply contract opportunities and the remaining two represent new demand contracts that need to be evaluated. It is important to the company to decide if it is worth to invest in these contracts. As stated above, only thermoelectric dispatch (along with spot energy prices) was considered to be uncertain in this work. Scenarios for thermoelectric dispatch were obtained by the official Brazilian energy planning program described in [130].

All computations were carried out on a Intel Xeon X5650 2.67GHz, with 2 processors, 48 Gbyte of RAM Memory, running 64 bit Windows Server 2008. All implementations were performed using AIMMS 3.13 and Gurobi 5.1 to solve resultant linear problems.

4.3.2 Solving smaller instances of the problem

We solved the original problem 100 times considering 10 scenarios randomly chosen from a reference sample with 200 scenarios. Then the probability of having profit losses was estimated as follows. Let $\bar{x}_{10}$ denote a solution to problem (4.3.1) with 10 scenarios randomly chosen, and $f_{200}(\cdot)$ denote the objective function in problem (4.3.1) considering the reference sample with $N = 200$ scenarios. We denote by f^*_{200} the optimal value of the latter instance (this value was obtained by employing a decomposition technique, as described in § 4.4 below). We estimate the probability of having profit losses greater or equal to $\ell \geq 0$ by

$$\mathbb{P}\Big(f_{200}(\bar{x}_{10}) - f^*_{200} \geq \ell\Big).$$

Figure 4.3 shows on the x-axis the profit loss ℓ and on the y-axis the above probability estimate. The

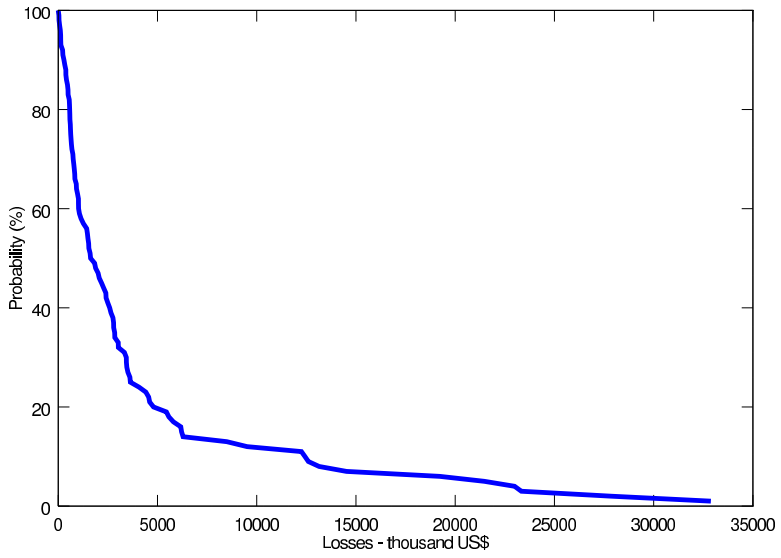


Figure 4.3: Probability of profit losses

figures shows that there is a significant probability – about 20% – of having, for instance, US$ 5 million

or more of profit loss. Some statistics on these losses, such as the average, standard deviation, minimum and maximum values are presented in Table 4.1. We can conclude that choosing randomly a small

Statistics	(10^3 US$)
Average	3 938.26
Std Deviation	6 168.83
Min	0
Max	32 881.94

Table 4.1: Statistics on objective function values

sample with 10 scenarios can lead to losses of up to approximately US$ 33 million. On average, the losses are around US$ 4 million. Thereby, we conclude that: (i) uncertainties play an important role in the Brazilian natural gas planning problem; and (ii) a smaller sample of scenarios should not be chosen randomly.

In order to represent the uncertainties in a better manner we shall consider larger samples of scenarios. However, for the considered application and $N = 80$ gas demand scenarios, the LP (4.3.1) has approximately $20,883,165$ variables and $6,752,380$ constraints; being too large to be solved by traditional LP solvers even using a powerful computer like the one described in § 4.3.1. The design premise of most commercially available solvers requires in-memory storage of the problem matrix and any disk caching can drastically reduces performance. In this manner, for larger instances where it is impossible to allocate RAM memory for the whole problem the best commercials solvers will still perform poorly. If increasing the available computer memory is not an option, then for solving the problem with $N \geq 80$ scenarios one may consider two approaches: (a) decomposition; (b) scenario reduction. In the remaining of this work we investigate these two alternatives, starting from approach (a).

4.4 Decomposition

We have concluded in the previous section that representing the uncertainties on the gas demand is an important matter for the considered natural gas network planning problem. However, when the number of scenarios is greater than 80 the problem becomes too large and can not be solved by commercial LP solvers. In this section we describe a decomposition technique applied with a nonsmooth optimization method for solving problem (4.3.1).

4.4.1 Two-stage stochastic linear programming formulation

Following the lead of [181], the linear problem (4.3.1) can be decomposed as

$$\min_{x} \; f(x) \;\; \text{s.t.} \;\; x \in X \quad \text{with} \;\; f(x) := \langle c, x \rangle + \sum_{i=1}^{N} \pi_i Q(x, \omega_i), \tag{4.4.1}$$

where $\sum_{i=1}^{N} \pi_i Q(x, \omega_i)$ is the expectation of the second stage costs given by

$$Q(x, \omega) := \begin{cases} \min\limits_{y} & \langle q, y \rangle \\ \text{s.t.} & Wy \leq h(\omega) - Tx \\ & y \in Y. \end{cases} \tag{4.4.2}$$

Given the characteristics of problem (4.2.1), problem (4.4.2) has a solution for every given investment decision $x \in X$, and for all (gas demand) scenarios ω_i, $i = 1, \ldots, N$ (the company can always buy gas in the international market to supply a domestic demand). This property is known in the stochastic

programming literature as *relatively complete recourse*. Given this assumption, it is well known that problem (4.4.1)-(4.4.2) is convex, finite valued, but nonsmooth; see ([181, § 2.1]) for further information.

Most optimization techniques for solving nonsmooth problems rely on an oracle (black-box) to provide first-order information of f. For the considered two-stage stochastic linear problem (4.4.1)-(4.4.2), an oracle is a decomposable optimization procedure that computes, for any given feasible point x, the value of the objective function and a subgradient at this point. Such procedure is described below:

Oracle 4.4.1. *(Oracle for two-stage stochastic linear problems).*

▷ *Step 0*

Input: $x_k \in X$

▷ *Step 1*

for $i = 1, 2, \ldots, N$ ***do***
    Solve problem (4.4.2) *for* $x = x_k$ *and* $\omega = \omega_i$
    Get the optimal value $Q(x_k, \omega_i)$
    Obtain a Lagrange multiplier u_i *for* (4.4.2)
end for

▷ *Step 2*

Compute the value of the function:
$f(x_k) \leftarrow \langle c, x_k \rangle + \sum_{i=1}^{N} \pi_i Q(x_k, \omega_i)$
Compute a subgradient:
$g_k \leftarrow c - \sum_{i=1}^{N} \pi_i T^\top u_i$

Output: $(f(x_k), g_k)$.

Proposition 2.2 in [181] ensures that g_k computed above is indeed a subgradient of f at the point x_k, i.e.,

$$f(x) \geq f(x_k) + \langle g_k, x - x_k \rangle \quad \text{for all} \quad x \in \mathbb{R}^n .$$

In the following section we present the optimization tool employed in this work to solve problem (4.4.1)-(4.4.2), making use of Oracle 4.4.1.

4.4.2 Proximal bundle method

Bundle methods are designed to solve nonsmooth convex optimization problems by making use of only first-order information of the involved functions; see [94] and [29]. Such methods are well known by their robustness and by having an efficient stopping test. Moreover, differently from cutting-plane methods like L-Shaped [186], most bundle methods have limited memory: the bundle of oracle information can be kept bounded saving computational memory without impairing convergence. This is particularly interesting for large-scale optimization problems, as the considered one.

This section restricts itself to the presentation of a proximal bundle algorithm suitable for solving problem (4.4.1)-(4.4.2). We refer to [174], [65], [66], and [148] for other bundle method variants.

4.4.2.1 Description of the method

The method generates a sequence of feasible iterates $\{x_k\} \subset X$. For each point x_k, Oracle 4.4.1 is called to compute $f(x_k)$ and a subgradient g_k. With such information, the method creates the linearization

$$\bar{f}_k(x) := f(x_k) + \langle g_k, x - x_k \rangle \quad (\leq f(x)) .$$

At iteration k a polyhedral *cutting-plane* model of f is available:

$$\check{f}_k(x) := \max_{j \in J_k} \bar{f}_j(x) \quad \text{with} \quad J_k \subset \{1, \ldots, k\} . \tag{4.4.3}$$

The set $\{x_j,\, f(x_j), g_j\}_{j\in J_k}$ is called information bundle. For instance, the L-Shaped method [186] takes $J_k = \{1,\dots,k\}$ for all k. In contrast, bundle methods can keep J_k with only two properly chosen linearizations, as shown in Remark 4.4.2 below.

Given a parameter $t_k > 0$ and denoting $\hat{x}_k$ a stability center (the best past iterate) at iteration k, the next iterate x_{k+1} is the unique solution to the quadratic program

$$\min_{x\in X} \check{f}_k(x) + \frac{1}{2t_k}\left\|x - \hat{x}_k\right\|^2, \tag{4.4.4}$$

which is equivalent to

$$\left\{\begin{array}{ll} \min\limits_{x,r} & r + \frac{1}{2t_k}\left\|x-\hat{x}_k\right\|^2 \\ \text{s.t.} & \bar{f}_j(x) \le r,\ \forall\, j\in J_k \\ & x\in X,\ r\in\mathbb{R}\,. \end{array}\right. \tag{4.4.5}$$

Denoting by i_X the indicator function of the set X, the optimality conditions for (4.4.4) give (see [43] for more details)

$$x_{k+1} = \hat{x}_k - t_k\hat{g}_k, \tag{4.4.6}$$

with

$$\hat{g}_k = p_f^k + p_X^k \quad\text{and}\quad \left\{\begin{array}{ll} p_f^k := \sum\limits_{j\in J_k}\alpha_j^k g_j & \in \partial\check{f}_k(x_{k+1}) \\ p_X^k := -\dfrac{x_{k+1}-x_k}{t_k} - p_f^k & \in \partial i_X(x_{k+1})\,. \end{array}\right. \tag{4.4.7}$$

The simplicial multiplier α^k satisfies the following relations for all $j\in J_k$

$$\sum_{j\in J_k}\alpha_j^k = 1\,,\quad \alpha_j^k \ge 0\,,\quad \alpha_j^k[\check{f}_k(x_{k+1}) - \bar{f}_j(x_{k+1})] = 0,$$

and can be used to save storage without impairing convergence. More precisely, "inactive" indices, corresponding $\alpha_j^k = 0$, can be dropped: $J_{k+1} \supset \{j\in J_k :\, \alpha_j^k \neq 0\}$.

An important convergence parameter of the method is the following:

$$\phi_k := f(\hat{x}_k) - \check{f}_k(x_{k+1}) - t_k\left\|\hat{g}_k\right\|^2 + \langle \hat{g}_k, \hat{x}_k\rangle\,. \tag{4.4.8}$$

It can be shown (see [149, Thm. 4.5]) that

$$f(\hat{x}_k) \le f(x) + \phi_k - \langle \hat{g}_k, x\rangle \quad \forall\, x\in X\,.$$

Therefore, proximal bundle algorithm can stop with a satisfactory solution $\hat{x}_k$ when both ϕ_k and $|\hat{g}_k|$ are small. In fact, [149, Thm 4.5] ensures that convergence amounts to obtaining the following property:

$$\text{a subsequence } \{(\phi_k,\hat{g}_k)\}_{k\in K} \text{ converges to } (\phi,0) \text{ with } \phi\le 0.$$

A *rule* decides whether to move the center ($\hat{x}_{k+1} = x_{k+1}$, descent-step) or to keep it ($\hat{x}_{k+1} = \hat{x}_k$, null-step). This rule views $f(\hat{x}_k)$ as a threshold, which each iteration strives to improve: a descent-step is performed if it improves the threshold by a definite amount; say

$$f(x_{k+1}) \le f(\hat{x}_k) - \kappa\,\mathtt{v}_k, \tag{4.4.9}$$

with $\mathtt{v}_k := f(\hat{x}_k) - \check{f}_k(x_{k+1})$ and some fixed $\kappa\in(0,1)$.

Algorithm 6 outlines the considered proximal bundle method variant. If (x_{k+1}, r_{k+1}) is a solution to problem (4.4.5), then $r_{k+1} = \check{f}(x_{k+1})$. Hence, the updating rule of $\mathtt{v}_k$ on line 6 of the algorithm coincides with the definition in (4.4.9).

Algorithm 6 Proximal bundle method

▷ Step 0: initialization

Select $\kappa \in (0,1)$ and $t_1 \geq t_{\min} > 0$
Choose $x_1 \in X$ and stopping tolerances, Tol_ϕ, $\text{Tol}_g > 0$
Call Oracle 4.4.1 to compute $(f(x_1), g_1)$ and set $\hat{x}_1 \leftarrow x_1$ and $J_1 \leftarrow \{1\}$,
for $k = 1, 2, \ldots$ **do**

▷ Step 1: Next iterate

    Obtain (x_{k+1}, r_{k+1}) and α^k by solving (4.4.5)
    Set $\hat{g}_k \leftarrow (\hat{x}_k - x_{k+1})/t_k$, $\mathtt{v}_k \leftarrow f(\hat{x}_k) - r_{k+1}$, and ϕ_k as in (4.4.8).

▷ Step 2: Stopping test

    if $\phi_k \leq \text{Tol}_\phi$ and $\|\hat{g}_k\| \leq \text{Tol}_g$) **then**
        return $\hat{x}_k$ and $f(\hat{x}_k)$
    end if

▷ Step 3: Oracle call

    Call Oracle 4.4.1 to compute $(f(x_{k+1}), g_{k+1})$
    Define $t_{\mathtt{aux}} = 2t_k[1 + (f(\hat{x}_k) - f(x_{k+1}))/\mathtt{v}_k]$
    if $f(x_{k+1}) \leqslant f(\hat{x}_k) - \kappa \mathtt{v}_k$ **then**
        Set $\hat{x}_{k+1} \leftarrow x_{k+1}$ and $t_{k+1} = \min\{t_{\mathtt{aux}}, 10t_k\}$
    else
        Set $\hat{x}_{k+1} \leftarrow \hat{x}_k$ and $t_{k+1} = \min\{t_k, \max\{t_{\mathtt{aux}}, t_{\min}\}\}$
    end if

▷ Step 4: Bundle management

    Choose $J_{k+1} \supset \{j \in J_k : \alpha_j^k \neq 0\} \cup \{k+1\}$
end for

Remark 4.4.2 (Bundle compression)**.** *It is worth mentioning that the set J_k gathering bundle information can be kept with at most $M_{\max}$ indices, for some chosen integer $M_{\max} \geq 2$. In fact, if at Step 4 of Algorithm 6 one has $|\{j \in J_k : \alpha_j^k \neq 0\}| = M_{\max}$, one can choose any two indices $j, i \in J_k$ and replace the two triples $(x_j, f(x_j), g_j)$ and $(x_i, f(x_i), g_i)$ by the artificial one $(x_{k+1}, \check{f}_k(x_{k+1}), \hat{g}_k)$. In this manner, one of the indices j or i will be related to the triple $(x_{k+1}, \check{f}_k(x_{k+1}), \hat{g}_k)$, while the other one can be eliminated from the set J_k. Since the bundle updating incorporates the new index $k+1$ into J_{k+1}, the bundle size will remain $M_{\max}$. This strategy is called* bundle compression*, and it is an efficient manner to keep the auxiliary problem* (4.4.5) *easy to solve without impairing convergence of the algorithm.*

Another proximal bundle method variant designed for two-stage stochastic linear programming is the so called *regularized decomposition*, proposed by [174]. Algorithm 6 is more general than the regularized decomposition in the sense that the presented algorithm uses the bundle compression mechanism and updates the prox-parameter t_k along the iterative process. Convergence analysis of Algorithm 6 can be obtained in [94], and in a more general setting in [149].

4.4.3 Numerical assessment: decomposition and bundle method

Formulation (4.3.1) becomes computationally intractable for 80 gas demand scenarios or more. To solve the deterministic equivalent problem using $N = 200$ scenarios we then used Algorithm 6 and Oracle 4.4.1.

With 200 scenarios, the corresponding problem has more than 52 million variables and almost 17 million constraints, and it was solved in 79 iterations for tolerances $\text{Tol}_\phi = 10^{-3}\sqrt{n}$ and $\text{Tol}_g = 10^{-3}\sqrt{n}$, with $\sqrt{n} = 684$ (n is the dimension of the first-stage variable x, i.e., investment decisions).

Figure 4.4 shows the evolution of $\{-f(x_k)\}$ along iterations 10 to 79. Circular markers indicate iterations at which a descent-step was found. We recall that for the considered application, minimizing the objective function in problem (4.3.1) corresponds to maximizing the total profit of the company: $-f(x)$ is the profit provided by the plan of investment x. Algorithm 6 using Oracle 4.4.1 was able to solve the deterministic

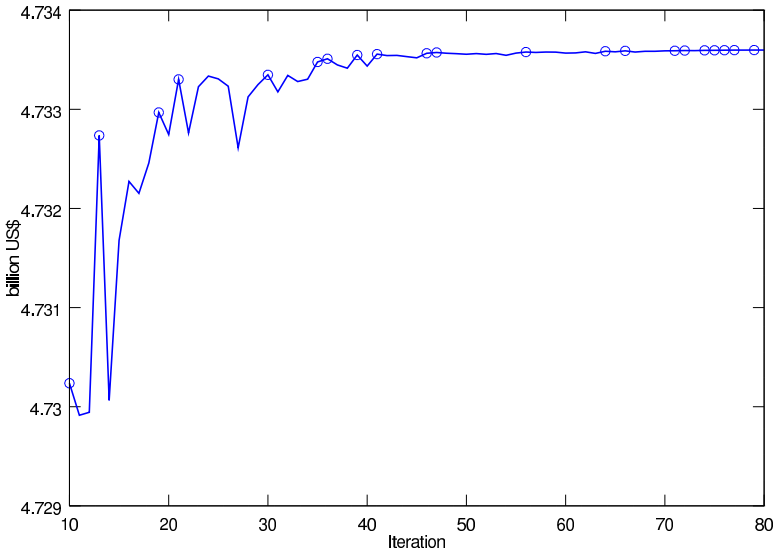


Figure 4.4: Evolution of $-f(x^k)$, $k = 10, \cdots, 79$.

equivalent problem, but it took 45 hours and 48 minutes in order to satisfy the stopping criteria. It turns out that each call to Oracle 4.4.1 takes around 34 minutes. Roughly speaking, allocating computational memory and solving problem (4.4.2) for a given x and ω takes 10 seconds of CPU time. Therefore, supposing problem (4.3.1) with $N = 2,000$ scenarios and estimating 70 iterations for Algorithm 6 to stop, the total CPU time for solving the resulting problem would be more than 15 days. In a strategic level of planning, as it is considered in this work, this amount of CPU time may be acceptable. However, other strategies should be tested as an attempt to reduce solving time while still considering uncertainties in a proper manner.

4.5 Optimal scenario reduction

In this section we apply the scenario reduction technique proposed in [80] and [82] to efficiently select, among N demand gas scenarios $\{\omega_1, \omega_2, \ldots, \omega_N\}$ with probability π_i for $i = 1, \ldots, N$, a smaller but representative subset of scenarios. Reducing the number of scenarios entails redistributing the initial set of probabilities $P := \{\pi_i : i = 1, \ldots, N\}$ by taking an index set $I_{\texttt{rep}} \subset \{1, 2, \ldots, N\}$ such that:

$$\tilde{P} := \{\tilde{\pi}_i : i = 1, \ldots, N\} \quad \text{with} \quad \begin{cases} \tilde{\pi}_i = 0 \text{ for } i \notin I_{\texttt{rep}} \\ \sum_{i \in I_{\texttt{rep}}} \tilde{\pi}_i = 1\,. \end{cases}$$

With such a redistribution $\tilde{P}$, the resulting (smaller) deterministic equivalent problem is

$$\min_{x \in X} \tilde{f}(x) \quad \text{with} \quad \tilde{f}(x) = \langle c, x \rangle + \sum_{i \in I_{\texttt{rep}}} \tilde{\pi}_i Q(x, \omega_i)\,. \tag{4.5.1}$$

Let $\texttt{VAL}(P)$ and $\texttt{VAL}(\tilde{P})$ be the optimal values for the problems (4.4.1) and (4.5.1), respectively. Accordingly, $S(P) \subset X$ and $S(\tilde{P}) \subset X$ are the respective solution sets (it is assumed that both $S(P)$ and $S(\tilde{P})$ are nonempty sets). Let the number $N_{\texttt{rep}} = |I_{\texttt{rep}}|$ of representative scenarios be given (naturally, $N_{\texttt{rep}} < N$). What we wish is to determine $\tilde{P}$ such that the distance between the optimal values $|\texttt{VAL}(P) - \texttt{VAL}(\tilde{P})|$ and the error $\epsilon \geq 0$ such that $S(\tilde{P}) \subset S(P) + \epsilon B(0,1)$ are as small as possible (here $B(0,1)$ stands for the unit ball in $\mathbb{R}^n$). In order to accomplish this task the work [82] proposes to use

a probabilistic metric that estimates the distance between the optimal values, and allows at same time for stability of the solution sets, i.e., a small $\epsilon \geq 0$.

Given $N_{\mathtt{rep}}$, the best redistribution $\tilde{P}$ of P is determined by solving a combinatorial optimization problem, as shown below. The result in [80, Thm. 2] shows that given a norm or pseudonorm $d : \Omega \times \Omega \to \mathbb{R}_+$ (for instance, $d(\omega_i, \omega_j) = \|\omega_i - \omega_j\|_2$ can be the Euclidean distance) and a set $I \subseteq \{1, \ldots, N\}$, the particular case of the Monge-Kantorovich functional

$$\mathtt{MK}(I) := \sum_{j \notin I} p_j \min_{i \in I} d(\omega_j, \omega_i)$$

is, except for a multiplicative constant, an upper bound for the distance between the given probability P and the new one $\tilde{P}$, defined by

$$\tilde{\pi}_i = \begin{cases} \pi_i + \sum_{j \in J_i} \pi_j & \text{if } i \in I \\ 0 & \text{if } i \notin I \end{cases} \quad \text{where } \; J_i := \{j \notin I : i \in \arg\min_{l \in I} d(\omega_j, \omega_l)\}\,. \tag{4.5.2}$$

Providing that the feasible set X is compact, it follows from [80, Eq. (4) and Thm. 2] that

$$\left| \sum_{i=1}^{N} \pi_i Q(x, \omega_i) - \sum_{i \in I} \tilde{\pi}_i Q(x, \omega_i) \right| \leq L \cdot \mathtt{MK}(I) \quad \text{for each } x \in X\,,$$

where $L > 0$ is a constant that depends on the problem and on the chosen pseudonorm d. Given the rule (4.5.2) for the new probability $\tilde{P}$, a natural criterion for approximating problem (4.4.1) by problem (4.5.1) should strive to minimize the functional $\mathtt{MK}(I)$, by choosing the best set $I = I_{\mathtt{rep}} \subset \{1, \ldots, N\}$ with $N_{\mathtt{rep}}$ indices. This is a combinatorial optimization problem, and thus difficult to solve. Following [82], the representative scenario index set $I_{\mathtt{rep}}$ is chosen by a heuristic method, called *Fast Forward Selection*. More precisely, this work applies Algorithm 2.4 proposed in [82] to select representative scenarios of gas demand. The main idea of this algorithm is to iteratively solving problems of the form

$$\min_{I \subset \{1,2,\ldots,N\}} \mathtt{MK}(I) \;\; \text{s.t.} \;\; |I| = N - i\,,$$

for $i = N, N-1, \ldots, N - N_{\mathtt{rep}}$. We refer to [80] and [82] for more information on scenario reduction for two-stage stochastic programming; see also [83], [52] and [158] for the multistage setting.

4.5.1 Numerical assessment: optimal scenario reduction

By using optimal scenario reduction technique, a subset $\Omega_{N_{\mathtt{rep}}} = \left\{\omega_{j_1}, \omega_{j_2}, \cdots, \omega_{j_{N_{\mathtt{rep}}}}\right\}$ is chosen from the set of $N = 200$ scenarios used in § 4.4. Then, the reduced deterministic equivalent problem (4.5.1) is solved, using only $N_{\mathtt{rep}}$ selected scenarios $\omega \in \Omega_{N_{\mathtt{rep}}}$ with new probability $\tilde{P}$. As usually $N_{\mathtt{rep}} \ll N$, it takes significantly less CPU time to solve (4.5.1) than to solve problem (4.4.1).

We have considered 7 different instances of problem (4.5.1) corresponding to taking

$$N_{\mathtt{rep}} \in \{10,\ 20,\ 30,\ 40,\ 50,\ 60,\ 70\}.$$

Let $\bar{x}_{N_{\mathtt{rep}}}$ be an optimal investment decision to problem (4.5.1) and

$$\tilde{f}(\bar{x}_{N_{\mathtt{rep}}}) = \langle c, \bar{x}_{N_{\mathtt{rep}}} \rangle + \sum_{i \in I_{\mathtt{rep}}} \tilde{\pi}_i Q(\bar{x}_{N_{\mathtt{rep}}}, \omega_i)$$

be its optimal value. In Table 4.2 we compare first stage (investment) cost, $\langle c, \bar{x}_{N_{\mathtt{rep}}} \rangle$, *expected* second stage costs

$$\sum_{i \in I_{\mathtt{rep}}} \tilde{\pi}_i Q(\bar{x}_{N_{\mathtt{rep}}}, \omega_i) \quad \text{and} \quad \sum_{i=1}^{200} \pi_i Q(\bar{x}_{N_{\mathtt{rep}}}, \omega_i)\,, \tag{4.5.3}$$

$N_{\texttt{rep}}$	Obj. Function (10^6 US$)			Problem Size		CPU (s)
	$\langle c, \bar{x}_{N_{\texttt{rep}}} \rangle$	$\sum_{i \in I_{\texttt{rep}}} \tilde{\pi}_i Q(\bar{x}_{N_{\texttt{rep}}}, \omega_i)$	$\sum_{i=1}^{200} \pi_i Q(\bar{x}_{N_{\texttt{rep}}}, \omega_i)$	Variables	Constraints	
10	-8.69	4 367.07	4 733.38	2 610 995	845 080	151
20	-8.70	4 473.30	4 733.54	5 221 305	1 688 980	260
30	-8.72	4 515.01	4 733.60	7 831 615	2 532 880	471
40	-8.72	4 566.87	4 733.60	10 441 925	3 376 780	560
50	-8.72	4 597.55	4 733.60	13 052 235	4 220 680	843
60	-8.72	4 645.87	4 733.60	15 662 545	5 064 580	1 025
70	-8.72	4 692.17	4 733.60	18 272 855	5 908 480	1 207
200	-8.72	4 733.60	4 733.60	52 206 885	16 879 180	164 879

Table 4.2: Problem comparison for $N_{\texttt{rep}} \in \{10, 20, 30, 40, 50, 60, 70\}$

number of variables and constraints, and CPU time for each instance $N_{\texttt{rep}}$. As expected, increasing $N_{\texttt{rep}}$ increases solution $\bar{x}_{N_{\texttt{rep}}}$ quality. Moreover, as reported in Table 4.2 the values in (4.5.3) become closer to each other as $N_{\texttt{rep}}$ increases, showing effectiveness of the scenario reduction technique.

In many practical applications, decision makers are concerned not only with optimal values deviation, but also with optimal decision variables deviation. The practical interest is the *optimal policy* - that is, the best first stage decisions. In our application, deviation on optimal solutions might result in very different company's actions: for instance, expanding a pipeline rather than importing gas. Figure 4.5 shows different decisions for a specific investment project when solving the problem with different number of scenarios. We see that the bigger the number $N_{\texttt{rep}}$ of scenarios the closer are the optimal

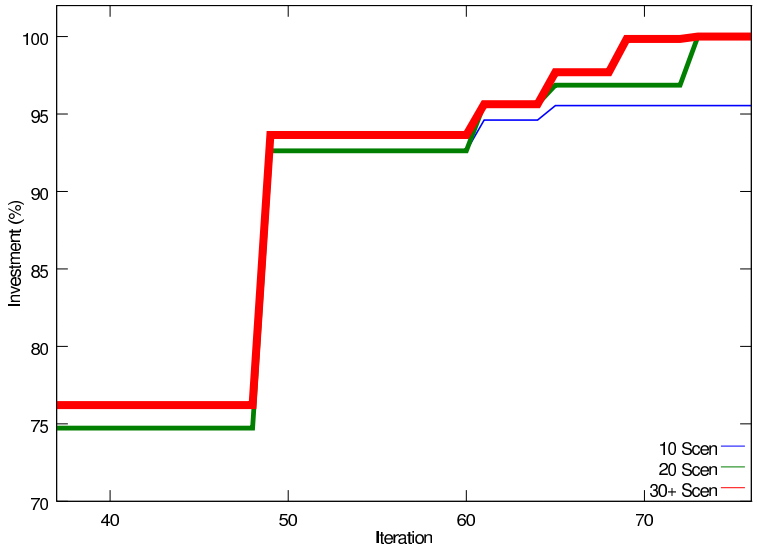


Figure 4.5: Different investment decisions

decisions (obtained with $N = 200$ scenarios). By using $N_{\texttt{rep}} = 30$ scenarios or more, optimal decisions are achieved for this investment project.

4.6 Concluding remarks

In this work we have investigated the Brazilian natural gas network planning problem by considering stochasticity in the gas demand. Numerical results show that a stochastic setting to the problem should be considered due to the potential improvements in investment policy and net profits. The gain obtained by using a two-stage stochastic linear approach to the problem was estimated to be approximately US$ 38 million. Such estimate was drawn by comparing the obtained investment decisions to the ones produced by a deterministic model that ignores uncertainties in the problem.

We have shown that unstable optimal values and investment decisions can be obtained if small samples of scenarios ($N = 10$) are randomly chosen to represent the gas demand. On the other hand, solving the problem for larger samples, say 80 scenarios or more, is only possible via decomposition technique (and specialized nonsmooth optimization methods). For $N = 200$ gas demand scenarios the resulting problem has more than 52 million variables and almost 17 million constraints. CPU time needed to solve the problem was almost 46 hours, and estimates of solving time for the case with $2,000$ scenarios is 15 days. The method used to solve large instances of the decomposed problem was the (state of the art) proximal bundle algorithm, presented with details in § 4.4.2.

Since the CPU time required to solve the problem with many scenarios (say $N = 2,000$) might not be affordable for the company's studies, an efficient tactic consists in combining both optimal scenario reduction and decomposition strategies: first, a representative but smaller subset of scenarios is select as described in § 4.5 and then the resulting planning problem is solved either by employing a LP commercial solver to the deterministic equivalent formulation (if the number of selected scenarios is small enough) or by the presented bundle method algorithm combined with decomposition.

To conclude we mention that the techniques presented in this work are implemented in the official model of PETROBRAS to assist decision making in this strategic supply chain problem, which is of significant importance for the economical development of Brazil.

Chapter 5

Convexity and optimization with copulæ structured probabilistic constraints

It is generally asserted that probability constraints are not "convex". However what is really meant is the convexity of the feasible set

$$\{x \in X : \mathbb{P}[G(x,\omega) \geq 0] \geq p\},$$

where $G : \mathbb{R}^n \times \Omega \to \mathbb{R}^m$ is a given mapping, $\omega : \Omega \to \mathbb{R}^m$ is a m-dimensional random vector defined on some probability space $(\Omega, \mathcal{A}, \mathbb{P})$, and $p \in (0,1]$ is a pre-specified probability level.

In probabilistically constrained optimization it is well known that the above set is convex whenever G is jointly quasi-concave and $\omega \in \mathbb{R}^m$ is a random vector having a density that itself admits generalized concavity properties. Important special cases involve structures as: $G(x,\omega) := \mathfrak{g}(x) - \omega$, wherein $\mathfrak{g}$ is concave. Then as soon as ω is chosen among an appropriate class of distributions, the function $\mathbb{P}[\omega \leq \mathfrak{g}(x)]$ can be asserted to have some degree of generalized concavity, which in turn entails convexity of its level sets. Convenient choices for ω are multivariate Gaussian, multivariate log-normal, multivariate student. Situations readily occur wherein this general result can not be employed: for instance, when the mapping $\mathfrak{g}$ has generalized concavity properties only.

Despite what is commonly stated, it turns out that some degree of convexity remains preserved anyway, as investigated by R. Henrion and C. Strugarek in [90]. But instead of disposing of convexity results for all level sets of $\mathbb{P}[\omega \leq \mathfrak{g}(x)]$, it is only necessary in some applications to dispose of convexity results for p "large enough". In this chapter, the results of which are taken from

> W. van Ackooij and W. de Oliveira
> *Convexity and optimization with Copulae structured probability constraints*
> **Optimization**, *2016, volume 65, issue 7, pp. 1349-1376*,

the sufficiency of generalized concavity properties of $\mathfrak{g}$ in $G(x,\omega) := \mathfrak{g}(x) - \omega$ holding only on certain level sets of $\mathfrak{g}$ is highlighted. For that purpose, this chapter deals with Copulæ structured chance-constrained programs. The chapter investigates how potential "convexity" resulting from $\mathfrak{g}$ can be split from any "non-convexity" due to the combination of it with a probability constraint. To this end, it is employed a Generalized Benders' decomposition and a supporting hyperplane level bundle method algorithm to deal with the underlying optimization problem with generalized convexity properties is proposed.

5.1 Introduction

In this work we are interested in optimization problems involving separable probabilistic constraints of the type

$$\mathbb{P}[\omega \leq \mathfrak{g}(x)] \geq p. \tag{5.1.1}$$

Constraints of this form, also named *chance-constraints* or *probabilistic constraints* [166], express that the decision vector $x \in \mathbb{R}^n$ is feasible if and only if the random inequality system $\omega \leq \mathfrak{g}(x)$ is satisfied with high enough probability. These constraints are encountered in many engineering problems involving uncertain data. We can find applications in water management, telecommunications, electricity network expansion, mineral blending, chemical engineering etc. (e.g., [2, 7, 88, 138, 166]). For an overview of theory, numerics and applications of chance constraints we refer to [54, 166] and references therein.

It is worthy mentioning that separable probabilistic constraints of the form (5.1.1) are not the most general class, but still a class widely present in relevant applications; see for instance [2, 17] for applications on power system optimization and [129] for chance constrained optimization applied to transportation problems. Although non trivial, chance constraints of the form (5.1.1) are easier to handle than more general ones.

We will assume throughout this paper that each component ω_i of the random vector ω has a known unidimensional continuous distribution function $z_i \in \mathbb{R} \mapsto F_i(z_i) := \mathbb{P}[\omega_i \leq z_i]$, $i = 1, ..., m$. Therefore, Sklar's Theorem [184] ensures that constraint (5.1.1) can be represented by a composite function involving the mapping $\mathfrak{g}(x) = (\mathfrak{g}_1(x), \ldots, \mathfrak{g}_m(x))$, the marginal distributions F_i, $i = 1, ..., m$, and a copula $\mathfrak{C} : [0,1]^m \to [0,1]$:

$$\mathbb{P}[\omega \leq \mathfrak{g}(x)] = \mathfrak{C}(F_1(\mathfrak{g}_1(x)), ..., F_m(\mathfrak{g}_m(x)))\,. \tag{5.1.2}$$

Under this notation, the optimization problem we are interested in is:

$$\min_{x \in X(p)} \; f(x) \tag{5.1.3a}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function and, for a given $p \in (0, 1]$,

$$X(p) := \left\{ x \in X \;\middle|\; \begin{array}{l} \mathfrak{C}(F_1(\mathfrak{g}_1(x)), ..., F_m(\mathfrak{g}_m(x))) \geq p \\ \mathfrak{g}_i(x) \geq \ell_i,\; i = 1, \ldots, m \end{array} \right\}, \tag{5.1.3b}$$

with X a polyhedral set and $\ell \in \mathbb{R}^m$ a given vector.

An important matter for numerical tractability of chance-constrained optimization problems as (5.1.3) is convexity of its feasible set. In this work we will provide conditions under which $X(p)$ given in (5.1.3b) is a convex set. These conditions involve generalized concavity of the composite function $\mathfrak{C}(F_1(\mathfrak{g}_1(\cdot)), ..., F_m(\mathfrak{g}_m(\cdot)))$, a threshold $p^* \in (0, 1]$ and a condition of the type $p \geq p^*$. Different copulæ $\mathfrak{C}$ and different marginal distributions F_i provide different computable thresholds p^*.

The convexity results presented in this paper are essentially an extension of the work initiated in [3, 90, 91]. More specifically we show that all Archimedean copulæ belong to the class of δ-γ-concave copulæ introduced in [3]. In addition, we provide a stabilized variant of the supporting-hyperplane algorithm of [204], suitable for probabilistically constrained optimization problems with eventual convexity only. The new algorithm is combined with a generalized Benders decomposition that separates the potential non-convexity induced by the probability constraint from any inherent convexity of the nominal data. Prior to making this precise in Section 5.1.1 below, let us mention that [39] also considers copulæ in conjunction with probability constraints. This work, aiming to provide convexity results for probability constraints involving random technology matrices, seems to rely however on an unclear results on the independence of certain types of copulæ on the decision vector x ([39, Lemma 2.7]). This Lemma can be stated as follows:

Consider the probability function $\mathbb{R}^n \ni x \mapsto \phi(x) = \mathbb{P}[Tx \leq h]$, where the random $K \times n$ matrix T has rows following an elliptically symmetric distribution with positive definite covariance matrix. Then there exists a copula $\mathfrak{C}$ independent of x such that $\phi(x) = \mathfrak{C}(\Psi_1(g_1(x)), ..., \Psi_K(g_K(x)))$, where for each

$i = 1, ..., K$, $g_i : \mathbb{R}^n \to \mathbb{R}$ are deterministic mappings depending on $\mathfrak{g}_i$ and Ψ_i are x-independent marginal 1-dimensional distribution functions.

This Lemma is false:

Example 5.1.1. *Assume that the 2×2 matrix T follows a multivariate Gaussian distribution centered in 0 with correlation matrix (when T is seen as a "vector" stored rows first):*

$$S = \begin{bmatrix} 1 & 0.75 & -0.25 & -0.10 \\ 0.75 & 1.0 & -0.10 & -0.25 \\ -0.25 & -0.10 & 1.00 & 0.75 \\ -0.10 & -0.25 & 0.75 & 1.00 \end{bmatrix},$$

then Tx is a centered 2-variate Gaussian random variable. Moreover

$$\mathbb{P}[Tx \leq h] = F_{\tilde{\Theta}(x)}(g_1(x), g_2(x)),$$

where $\tilde{\Theta}(x)$ is a 2×2 correlation matrix having $\tilde{\Theta}(x)_{12} = \frac{-0.25x_1^2 - 0.25x_2^2 - 0.20x_1x_2}{x_1^2+x_2^2+1.5x_1x_2}$, $g_i(x) = \frac{\mathfrak{g}_i}{\sqrt{x_1^2+x_2^2+1.5x_1x_2}}$, $i = 1, 2$. (see, e.g., [1, proof of Theorem 3.2.4] for details on this derivation) and $F_{\tilde{\Theta}(x)}$ is the 2-variate Gaussian distribution function of a centered Gaussian vector with correlation matrix $\tilde{\Theta}(x)$. Now the Gaussian copula $C^{\tilde{\Theta}(x)}$ (defined as $C^{\tilde{\Theta}(x)}(u_1, u_2) := F_{\tilde{\Theta}(x)}(\Phi^{-1}(u_1), \Phi^{-1}(u_2))$, resulting from Sklar's theorem, gives with $u = (0.25, 0.25)$, $x = (0, 1)$ and $y = (0.75, 1)$:

$$C^{\tilde{\Theta}(x)}(u) = 0.0387 \neq 0.0431 = C^{\tilde{\Theta}(y)}(u).$$

Note that since 1-dimensional Gaussian distribution functions (denoted Φ above) are continuous, by [185, Corollary to Theorem 1], the Gaussian copula is the unique copula satisfying the requested representation. Hence this provides a concrete counterexample against the Lemma.

The algorithms considered in [39] (the same are considered in [99]) are based on an ad-hoc implementation of a cutting plane method (with a priori generation of cutting planes) and a method seemingly related to p-efficient point approaches with interpolation (inner approximation). The convergence to an optimal solution is not established and the numerical experiments conducted on 5 instances indicate that the methods fail to close the gap below 2% on average. In contrast we provide a general optimization framework based on level bundle methods, establish convergence and show the interest of the methods on 1500 instances with respect to a competing standard software. We also mention the work [212] that uses a notion akin to g-concavity as introduced by [192] in order to generalize convexity results of probabilistically constrained feasible sets in the separable case (see also [3, Remark 3.3]). However they consider the situation of components-wise independence for the random vector ω.

5.1.1 Separating out convexity

Let us introduce extra variables $u \in \mathbb{R}^m$ to reformulate problem (5.1.3) as follows:

$$v_{\min} := \min_{u \in [0,1]^m} \quad v(u) \quad \text{s.t.} \quad \mathfrak{C}(u) \geq p, \ \text{ and } \ u_i \geq F_i(\ell_i), \ i = 1, \ldots, m\,, \tag{5.1.4a}$$

with objective function given by

$$v(u) = \min_{x \in X} f(x) \ \text{ s.t. } \ \mathfrak{g}_i(x) \geq F_i^{-1}(u_i), \ i = 1, \ldots, m\,. \tag{5.1.4b}$$

The relation between formulations (5.1.3) and (5.1.4) is established in § 5.3.2. In the parlance of Bender's decomposition, subproblem (5.1.4b) is denoted by *slave problem* and (5.1.4a) is the so-called *master problem*. Notice that the optimization of the slave problem is performed only in variable x, i.e., u is seen in (5.1.4b) as a parameter. Therefore, solving (5.1.4b) for a given u amounts to solving a convex optimization problem provided that f is a convex mapping and $\mathfrak{g}_i$ disposes of generalized concavity properties. As a result, any potential non-convexity of problem (5.1.3) has been moved to the master problem. This motivates us to solve (5.1.3) through an algorithm based on Bender's decomposition.

5.1.2 Bender's decomposition: a bird's-eye view

Generalized Bender's decomposition (GBD) hinges on the key observation that certain problems can become significantly easier when a subset of variables is fixed. This observation originally made for problems with an underlying linear structure [26] potentially with stochastics [186] was generalized to problems with some underlying convexity in the seminal work by Geoffrion [76]. For a general overview we refer to [44, 69].

The optimal value of the slave problem is seen as a mapping of the fixed variables, called *slave mapping* or *value function*. The master problem is related to finding the optimal allocation of the previously temporarily fixed variables. It need not be a convex optimization problem. Since the domain of the value function need not be the whole space, the master problem is augmented with the so called *feasibility cuts* (i.e., an outer approximation of the convex domain of the value function). These can be computed using an auxiliary optimization problem involving slacks. The master problem is also enriched with *optimality cuts*, as a matter of fact, a cutting-plane model of the value function. In this view, GBD can be seen as a variant of the (Kelley's) cutting-plane method given in [103]. It can therefore be subject to the well-known oscillation effect and slow convergence. In order to tackle this, the authors of [213] suggest what we would now call an inexact lower oracle (see [149]), i.e., inexactly solving the slave problem to compute some inexact but "cheap" cuts. An appropriate choice of cuts (approximating the value function) is crucial, as illustrated in [108].

The works [60, 178, 182, 207, 211] concretely deal with strategies for generating good (optimality) cuts, occasionally with a problem-dependent flavour. The authors of [40] are concerned with generating strong feasibility cuts and provide an important contribution in this view. Further improvements to the general scheme are concerned with a relaxation of the convexity assumption of the slave problem by allowing for integer variables [35] or the use of generalized "logic" duality (called inference duality) in [98]. The success of GBD is easily seen from the existence of many application, e.g., [177] or [37, 67, 77, 132, 136, 140, 151, 170, 198] just to name a few. To the best of our knowledge only a single work [215] deals with the study of GBD under generalized concavity properties. The authors suggest a layered Benders decomposition framework (two embedded Generalized Benders decompositions).

In our setting, we cannot apply a cutting-plane method to solve the master problem because the Copula $\mathfrak{C}$ in (5.1.4a) need not be concave. However, due to the generalized concavity assumptions on $\mathfrak{C}$, and provided that $p \in (0, 1]$ is a large enough probability level (see Theorem 5.3.1 for more details), we rely on the supporting hyperplane method of [204] to solve problem (5.1.3) through Bender's decomposition.

5.1.3 Main contributions and organization of the work

In addition to theoretical results establishing δ-γ-concavity of all Archimedean copulæ (Theorem 5.3.3) and making explicit that it is sufficient for $\mathfrak{g}_i$ to dispose of generalized concavity properties on certain sets only (Theorem 5.3.1), we can enumerate the following contributions of this manuscript.

5.1.3.1 Contributions to supporting hyperplane and level bundle methods

In the present work, we rely on decomposition and employ a supporting hyperplane algorithms akin to [204] to find an optimal solution to problem (5.1.3) even when functions f and $\mathfrak{g}_i$ are nonsmooth. In contrast to [204] the proposed algorithm is able to handle extended real-valued objective functions. This is an important matter for dealing with value functions as in (5.1.4b). Another novelty with respect to [204] is that our algorithm employs a level-set regularization strategy similarly to [121] to avoid tailing-off effect that makes calculations unstable as the iteration process progresses. The algorithm is therefore an extension of level bundle methods [9, 121] to handle optimization problems having nonlinear constraints with generalized concavity properties and extended real-valued objective function. To the best of our knowledge, level bundle methods in the literature are only able to deal with: (a) linearly and nonlinearly constrained optimization problems involving convex functions, [121, 147]; or (b) linearly constrained

optimization problems with real-valued but quasi-convex objective function, [210]. In contrast to most level bundle methods in the literature that need to solve a QP master problem to define trial points, the proposed level bundle algorithm is general enough to define the master problem as a linear programming problem. This is an interesting feature when dealing with large-scale optimization problems. However, this feature might reduce convergence speed.

5.1.3.2 Contributions to the GBD literature

With respect to the existing literature on generalized Benders decomposition the contributions of the this work are as follows. We are concerned with local generalized concavity of mappings (generalized concavity of the mapping on certain level sets only). Compared to [215] we suggest a single layer framework. Moreover, we are interested in chance constrained problems with "left-hand side" uncertainty. To the best of our knowledge generalized Benders decomposition was not suggested for these problems before. With respect to this application, we incorporate in the master problem a special "feasible allocation" to avoid (possible) ill-conditioning related to stiff-gradients (see [134] and the discussion in [10]). The proposed algorithm combines in a single framework GBD, level bundle and supporting hyperplane methods.

Although we have not investigated the enhancements related to appropriately selecting elements in the value function sub-differential, in the spirit of [131], we believe that this feature can be appended to the framework in a straightforward manner.

5.1.3.3 Organization

This paper is organized as follows: in Section 5.2 we present notation and several (already known) concepts. Section 5.3 is dedicated to convexity results: (a) of the set $X(p)$ in (5.1.3b) provided p is large enough; (b) of the value function appearing in (5.1.4b). The algorithm and its convergence analysis are presented in Section 5.4. In Section 5.5 we approximate a very challenging chance-constrained optimization problem by adopting the framework of problem (5.1.3) and consider such a reformulated/approximated problem in our numerical experiments. We also consider chance-constrained optimization problems arising from cascaded reservoir management (with real-life data). A comprehensive battery of experiments is presented in the same section, where we have also solved problem (5.1.3) by the nonlinear optimization solver `IPOPT` [205].

5.2 Preliminaries: copulæ and generalized concavity

In this section we will review several of the key concepts required in the remainder of this paper.

5.2.1 Copulæ in a nutshell

In probabilistic terms, copulæ are parametrically specified joint probability distributions generated from given marginals distributions [141, 195]. In analytic terms, a copula is defined as follows [135]:

Definition 5.2.1. *A function $\mathfrak{C} : [0,1]^m \to [0,1]$ is called a copula if it satisfies the following conditions:*

(i) $\mathfrak{C}(1,\ldots,1,u,1,\ldots,1) = u$ *for all* $u \in [0,1]$*;*

(ii) $\mathfrak{C}(u_1,\ldots,u_{i-1},0,u_{i+1},\ldots,u_m) = 0$ *(the copula is zero if one of its arguments is zero);*

(iii) $\mathfrak{C}$ *is quasi-monotone on* $[0,1]^m$*.*

The last property can be equivalently expressed as the $\mathfrak{C}$-volume of any m-dimensional interval is non-negative (more details in [135]). A well known result is Sklar's theorem [184], which states that every multivariate cumulative distribution function F of a random vector $\omega \in \mathbb{R}^m$ with continuous marginals $F_i(y_i) = \mathbb{P}[\omega_i \leq y_i]$ can be written as

$$F(y_1, \ldots, y_m) = \mathfrak{C}(F_1(y_1), \ldots, F_m(y_m)) \,,$$

where $\mathfrak{C}$ is an appropriate copula. In order to give an idea of how Sklar's theorem works, we begin by recalling that $U_i = F_i(\omega_i)$ is uniformly distributed in the interval $[0, 1]$. By assuming that each marginal distribution function F_i is continuous, we have $F_i(y_i) \leq u_i$ if and only if $y_i \leq F_i^{-1}(u_i)$, for $i = 1, \ldots, m$. Consequently,

$$\begin{aligned} F(y_1, \ldots, y_m) = \mathbb{P}[\omega_1 \leq y_1, \ldots, \omega_m \leq y_m] &= \mathbb{P}[\omega_1 \leq F_1^{-1}(u_1), \ldots, \omega_m \leq F_m^{-1}(u_m)] \\ &= \mathbb{P}[U_1 \leq u_1, \ldots, U_m \leq u_m] \,. \end{aligned}$$

The copula of $\omega \in \mathbb{R}^m$ is defined as the joint cumulative distribution function of $(U_1, U_2, \ldots, U_m)$: $\mathfrak{C}(u) = \mathbb{P}[U_1 \leq u_1, \ldots, U_m \leq u_m]$, i.e., the m-copula $\mathfrak{C}$ above is a m-dimensional distribution function with all m univariate marginals being uniform in the interval $[0, 1]$.

We care to emphasize that Sklar's theorem is not "constructive". Hence, from a modeling perspective one would rather take a copula $\mathfrak{C}$ and through this choice implicitly fix the joint distribution (5.1.2). This will also be our angle of attack, i.e., we will assume that $\mathfrak{C}$ is given.

Several families of copulæ are known in the literature. We now focus on an important class, called Archimedean copulæ, which enjoy considerable popularity in a number of practical applications, (see references in [135]).

Definition 5.2.2. *A copula $\mathfrak{C}$ is called Archimedean if it has the representation*

$$\mathfrak{C}(u_1, \ldots, u_m) = \psi_\theta^{[-1]} \left(\psi_\theta(u_1) + \cdots + \psi_\theta(u_m) \right)$$

where $\psi_\theta : [0, 1] \to [0, \infty)$ is a continuous, strictly decreasing and convex function such that $\psi_\theta(1) = 0$ and θ is the real parameter on which it depends. The mapping ψ_θ is called the generator of the copula $\mathfrak{C}$.

The inverse of the generator function ψ_θ is written as ψ_θ^{-1}, and the pseudo-inverse $\psi_\theta^{[-1]}$ is given by

$$\psi_\theta^{[-1]}(t) = \begin{cases} \psi_\theta^{-1}(t) & \text{if} \quad 0 \leq t \leq \psi_\theta(0) \\ 0 & \text{if} \quad \psi_\theta(0) \leq t \leq \infty. \end{cases}$$

The following generators are commonly considered:

i) Joe's copula, with generator $\psi_\theta(t) = -\ln(1 - (1 - t)^\theta)$, $\theta \geq 1$;

ii) Frank's copula, with generator $\psi_\theta(t) = -\ln(\frac{e^{-\theta t} - 1}{e^{-\theta} - 1})$, $\theta \in \mathbb{R} \backslash \{0\}$;

iii) Ali-Mikhail-Haq's copula, with generator $\psi_\theta(t) = \ln(\frac{1-\theta}{t} + \theta)$, $\theta \in [-1, 1)$;

iv) Clayton copula, with generator $\psi_\theta(t) = \frac{1}{\theta} \left(t^{-\theta} - 1 \right)$, $\theta \in [-1, \infty) \backslash \{0\}$;

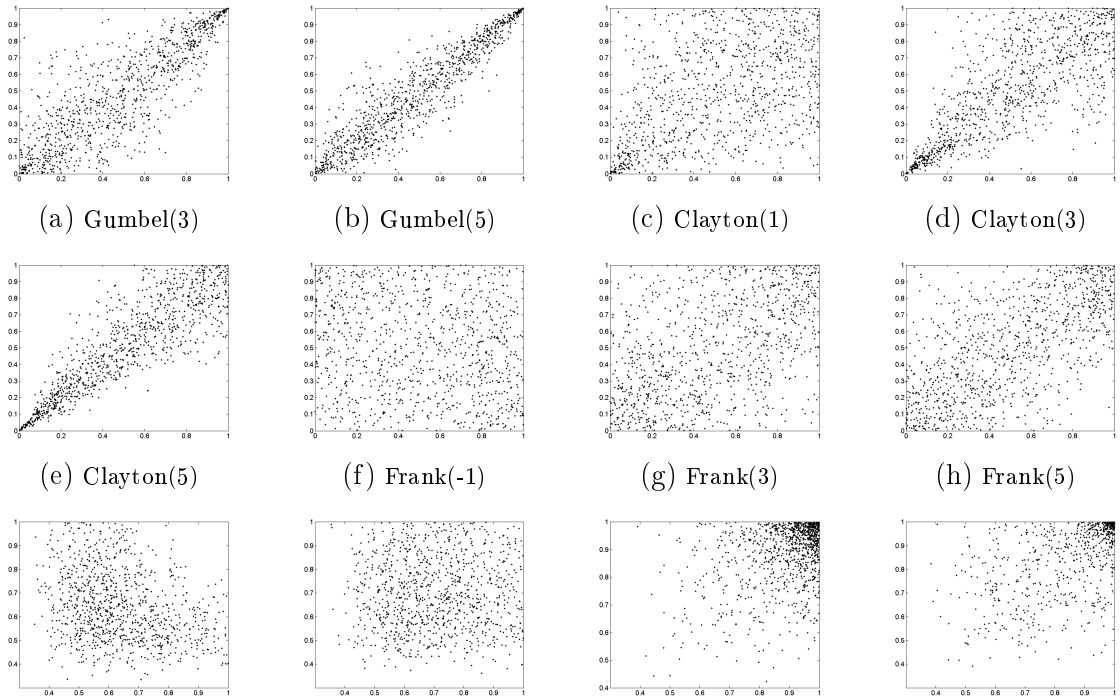
v) Gumbel copula, with generator $\psi_\theta(t) = (-\log(t))^\theta$, $\theta \in [1, \infty)$.

We provide a brief illustration of several dependency structures for $m = 2$ in Figure 5.1 (e.g., [152] and references therein). This figure illustrates a scatter plot of $u \in [0, 1]^2$ generated according to the joint distribution $\mathfrak{C}$.

In order to show that $X(p)$ in (5.1.3b) is a convex set for a given copula $\mathfrak{C}$, we will make extensive use of generalized concavity and its properties. We will introduce notation and useful results in the following subsection.

5.2.2 Generalized concavity and its properties

In order to define generalized concavity in a convenient way, the following function will be required.



(a) Gumbel(3) (b) Gumbel(5) (c) Clayton(1) (d) Clayton(3)

(e) Clayton(5) (f) Frank(-1) (g) Frank(3) (h) Frank(5)

(i) AMH(-1) (j) AMH(0) (k) AMH(0.9) (l) Joe(5)

Figure 5.1: The dependency structure for several copulæ. Each point represents a randomly generated realization. Since the marginal distributions of a copula are uniform, each random realization belongs to $[0,1]^2$. The number in between parentheses is the parameter of the copula. The abbreviation AMH stands for Ali-Mikhail-Haq.

Definition 5.2.3. *Let $\alpha \in [-\infty, \infty]$ and $m_\alpha : \mathbb{R}_+ \times \mathbb{R}_+ \times [0,1] \to \mathbb{R}$ be defined as follows*

$$m_\alpha(a, b, \lambda) = 0 \text{ if } ab = 0,$$

for $a > 0$, $b > 0$, $\lambda \in [0,1]$:

$$m_\alpha(a,b,\lambda) = \begin{cases} a^\lambda b^{1-\lambda} & \text{if} \quad \alpha = 0 \\ \max\{a,b\} & \text{if} \quad \alpha = \infty \\ \min\{a,b\} & \text{if} \quad \alpha = -\infty \\ (\lambda a^\alpha + (1-\lambda)b^\alpha)^{\frac{1}{\alpha}} & \text{otherwise.} \end{cases}$$

The following lemma, given in [54], will be used throughout this text.

Lemma 5.2.4. *([54, Lemma 4.8]) Let m_α be the mapping as defined in Definition 5.2.3. The mapping $\alpha \mapsto m_\alpha$ is nondecreasing and continuous.*

We now provide the definition of generalized concavity:

Definition 5.2.5. *A non-negative function f defined on some convex set $C \subseteq \mathbb{R}^n$ is called α-concave ($\alpha \in [-\infty, \infty]$) if and only if for all $x, y \in C, \lambda \in [0,1]$:*

$$f(\lambda x + (1-\lambda)y) \geq m_\alpha(f(x), f(y), \lambda), \tag{5.2.1}$$

where m_α is as in Definition 5.2.3.

Remark 5.2.6. *If, for some $\alpha \in [-\infty, \infty]$, f is α-concave, then it is also $\tilde{\alpha}$-concave for all $\tilde{\alpha} \leq \alpha$. A function f is 0-concave if its logarithm is concave. This is usually referred to as log-concavity. For $\alpha \neq 0$, $\alpha \in \mathbb{R}$, the function f is α-concave if either f^α is concave for $\alpha > 0$ or f^α is convex for $\alpha < 0$. In particular, if $\alpha = 1$ then f is simply concave, and if $\alpha = -\infty$ then f is quasi-concave.*

For some further calculus rules with α-concavity we refer to Theorems 4.19–4.23 of [54]. We also recall the definition of generalized concavity for copulæ, [3]:

Definition 5.2.7. *Let $\gamma \in \mathbb{R}$ be given, and let the set $D(\gamma)$ be defined as $D(\gamma) = [0,1]^m$ for $\gamma > 0$, $D(0) = (-\infty, 0]^m$ and $D(\gamma) = [1, \infty)^m$ for $\gamma < 0$. Let $\delta \in [-\infty, \infty]$ be equally given. We call a copula $\mathfrak{C} : [0,1]^m \to [0,1]$ δ-γ-concave if the mapping $u : D(\gamma) \mapsto \mathfrak{C}(u^{\frac{1}{\gamma}})$ is δ-concave, whenever $\gamma \neq 0$ and $u : D(0) \mapsto \mathfrak{C}(e^u)$ is δ-concave whenever $\gamma = 0$.*

In Definition 5.2.7 the generalized concavity properties need not hold on the full set $D(\gamma)$, but only on a specific subset of it. It is therefore also useful to introduce the following local version of δ-γ-concavity for copula (see [3]).

Definition 5.2.8. *Let $q \in (0,1)^m$ be some point and define the sets $D(q,\gamma)$ as follows $D(q,\gamma) = \prod_{i=1}^m [q_i^\gamma, 1]$ for $\gamma > 0$, $D(q,0) = \prod_{i=1}^m [\log(q_i), 0]$ and $D(q,\gamma) = \prod_{i=1}^m [1, q_i^\gamma]$ for $\gamma < 0$. We call a copula $\mathfrak{C} : [0,1]^m \to [0,1]$ δ-γ-q-concave if the mapping $u : D(q,\gamma) \mapsto \mathfrak{C}(u^{\frac{1}{\gamma}})$ is δ-concave, whenever $\gamma \neq 0$ and $u : D(q,0) \mapsto \mathfrak{C}(e^u)$ is δ-concave whenever $\gamma = 0$.*

Remark 5.2.9. *Notice that for $\delta = -\infty$ and $\gamma = 1$, i.e., $-\infty$-1-concavity of a copula $\mathfrak{C} : [0,1]^m \to [0,1]$ the notion of δ-γ-concavity is equivalent to ordinary quasi-concavity of the same function $\mathfrak{C}$ on $[0,1]^m$. This property yields convexity of the level sets $\{u \in [0,1]^m : \mathfrak{C}(u) \geq p\}$, for all $p \in [0,1]$ and conversely, as is well known.*

5.3 Convexity statements: convexity of the nominal problem and the value function

5.3.1 Convexity of the nominal problem

We now gather and extend some results given in [3] to establish convexity of the feasible set $X(p)$ in (5.1.3b).

Theorem 5.3.1. *([11, Theorem 3.1]). Let $\omega \in \mathbb{R}^m$ be a random vector with associated copula $\mathfrak{C}$, and let $\mathfrak{g}_i : \mathbb{R}^n \to \mathbb{R}$ be functions such that*

$$\mathbb{P}[\omega \leq \mathfrak{g}(x)] = \mathfrak{C}(F_1(\mathfrak{g}_1(x)), ..., F_m(\mathfrak{g}_m(x))),$$

where F_i is the marginal distribution function of random variable ω_i, for $i = 1, ..., m$. Assume that, for any $i = 1, ..., m$, we can find $\alpha_i \in \mathbb{R}$, such that the functions $\mathfrak{g}_i$ are α_i-concave and a second set of parameters $\gamma_i \in (-\infty, \infty], \hat{b}_i > 0$ such that either one of the following conditions holds:

i) $\alpha_i < 0$ and $z \mapsto F_i(z^{\frac{1}{\alpha_i}})$ is γ_i-concave on $(0, \hat{b}_i^{\alpha_i}]$

ii) $\alpha_i = 0$ and $z \mapsto F_i(\exp z)$ is γ_i-concave on $[\log \hat{b}_i, \infty)$

iii) $\alpha_i > 0$ and $z \mapsto F_i(z^{\frac{1}{\alpha_i}})$ is γ_i-concave on $[\hat{b}_i^{\alpha_i}, \infty)$,

where $i \in \{1, ..., m\}$ is arbitrary. If the copula is either δ-γ-concave or δ-γ-$F(\hat{b})$-concave for $\gamma \leq \gamma_i \leq \infty$, $i = 1, ..., m$, it holds that

a) the set $M(p) := \{x \in \mathbb{R}^n : \mathbb{P}[\omega \leq \mathfrak{g}(x)] \geq p\}$ is convex for all $p > p^M := \max_{i=1,...,m} F_i(\hat{b}_i)$;

b) if, in addition, each individual distribution function F_i, $i = 1, ..., m$ is strictly increasing, then convexity can moreover be derived for all $p \geq p^M$;

c) if $\alpha_i \geq 0$ and F_i is γ_i-concave everywhere, $i = 1, ..., m$, then the set $M(p)$ is convex for all $p \in [0, 1]$.

We now weaken the assumption on $\mathfrak{g}_i$: assume that there exists a vector $b \in \mathbb{R}^m$, such that $\mathfrak{g}_i$ is α_i-concave, on the level sets $D := \{x \in \mathbb{R}^n : \mathfrak{g}_i(x) \geq b_i, \ \forall\, i = 1, \ldots, m\}$ for $\alpha_i \in \mathbb{R}$. Moreover, the marginals F_i satisfy once again either one of the conditions i)-iii), but they are not necessarily strictly increasing. By defining $p^* := \mathfrak{C}(F_1(b_1), \ldots, F_m(b_m))$, it holds that

d) if $b \geq \hat{b}$, then the set $M(p) \cap D$ is convex for all $p \geq p^*$;

e) if $\ell_i \geq \max\{b_i, \hat{b}_i\}$, $i = 1, \ldots, m$, then the feasible set $X(p)$ in (5.1.3b) is a convex set for all $p \geq p^*$.

Corollary 5.3.2. *In the setting of Theorem 5.3.1, assume that $\alpha_i = 1$ and that F_i is γ_i-concave everywhere, where $\gamma_i \in (-\infty, \infty]$ for all $i = 1, \ldots, m$. If the copula is $\delta - \gamma$-concave, the feasible set $X(p)$ in (5.1.3b) is convex for all $p \in [0, 1]$ regardless of $\ell_i \in [-\infty, \infty)$, $i = 1, \ldots, m$.*

Note that the extension of the results of [3] contained in Theorem 5.3.1 resides in making explicit that it is sufficient for $\mathfrak{g}_i$ to be generalized concave on specific sets only.

An important assumption in Theorem 5.3.1 is that the considered copula: $\mathfrak{C} : [0, 1]^m \to [0, 1]$ needs to be δ-γ-concave. Next, we show that all Archimedean copulæ belong to the class of δ-γ-concave copulæ and, hence, provide eventually convexity of the feasible set $X(p)$ given in (5.1.3b).

Theorem 5.3.3. *([11, Theorem 3.3]). Let $\mathfrak{C} : [0, 1]^m \to [0, 1]$ be an Archimedean copula, and $\psi : (0, 1] \to [0, \infty)$ be its generator. Then $\mathfrak{C}$ is a $-\infty$-1-concave copula, i.e., a quasi-concave copula.*

Remark 5.3.4. *We care to note that stronger generalized concavity properties are known for the Clayton copula. Indeed, it is δ-0 concave for specific δ values depending on the parameter of its generator (see [3]). Similarly, the Gumbel, independent and maximum copulæ are 0-0-concave as shown in [90].*

5.3.2 Convexity of the value function in generalized Benders decomposition

If $X(p)$ in (5.1.3b) is a nonempty and convex set, and f is a convex function, then under a Slater type assumption, any KKT point for problem (5.1.3) is also an optimal solution. Note that the Slater assumption, i.e., the existence of x such that (5.1.1) is satisfied strictly is a mild assumption (for instance also needed for stability results [89]). If, moreover, all the involved functions are differentiable, then the task of solving (5.1.3) can be carried out by general purpose solvers for nonlinear and differentiable optimization. There are, however, situations in which more specialized algorithms are preferable for solving problem (5.1.3) through suitable reformulations. For instance, when the considered copula is

difficult to evaluate, as discussed in § 5.5.1.4 below. Another situation arises when some or all constraints $\mathfrak{g}_i$ are nonsmooth, but suitable reformulations exist that allow their handling with specialized algorithms. This case is addressed in the first part of Section 5.5.

The use of additional variables frequently plays an important role in the quest of efficiently solving optimization problems. Indeed, since the marginal distribution function F_i is monotonically nondecreasing, the constraint $\mathfrak{g}_i(x) \geq \ell_i$ in (5.1.3b) is equivalent to $F_i(\mathfrak{g}_i(x)) \geq F_i(\ell_i)$. Therefore, by adding the extra variables $u \in [0,1]^m$ problem (5.1.3) is equivalent, in terms of optimal value and feasible region for x, to

$$\left\{ \begin{array}{lll} \min\limits_{(x,u)\in X\times[0,1]^m} & f(x) & \\ \text{s.t.} & \mathfrak{C}(u) & \geq p \\ & F_i(\mathfrak{g}_i(x)) & \geq u_i, \quad i = 1,\ldots,m \\ & u_i & \geq F_i(\ell_i), \quad i = 1,\ldots,m\,. \end{array} \right. \tag{5.3.1}$$

The value function $v : [0,1]^m \to \mathbb{R} \cup \{\infty\}$ defined in (5.1.4b) is thus obtained by splitting variables x and u from the above problem, and representing the constraint $F_i(\mathfrak{g}_i(x)) \geq u_i$ by $\mathfrak{g}_i(x) \geq F_i^{-1}(u_i)$, avoiding the need of computing derivatives for the (very often implicit) function F_i. As already mentioned in the introduction, under appropriate assumptions on functions f and $\mathfrak{g}_i$, $i = 1,\ldots,m$, problem (5.1.4b) can be easily solved. For instance, if f and all $\mathfrak{g}_i$ are linear (respectively concave quadratic) functions, (5.1.4b) is a linear (respectively convex conic) programming problem.

The value function v allows us to split problem (5.3.1) into two subproblems: the slave subproblem (5.1.4b) and the master problem (5.1.4a). The relation between formulations (5.1.3) and (5.1.4) is established by the following result, whose elementary proof is omitted.

Lemma 5.3.5. *Assume that problem* (5.1.3) *admits an optimal solution. Then problem* (5.1.4) *also admits an optimal solution and both optimal values are identical. Moreover, if u^* is optimal for* (5.1.4a)*, and x^* optimal for problem* (5.1.4b)*, in which $u = u^*$, then x^* is optimal for* (5.1.3).

Since $\mathfrak{C}$ is a copula we cannot expect $\mathfrak{C}$ to be concave in general (see for example Theorem 5.3.3). Still $\mathfrak{C}$ may have generalized concavity properties such as δ-γ-concavity (Definition 5.2.7). Consequently, the feasible set in (5.1.4a) can not be properly approximated by using first-order linearizations of $\mathfrak{C}$, a standard procedure in convex (nonsmooth) optimization. If $\mathfrak{C}$ happens to be δ-γ-concave for some $\gamma \leq 1$, then the feasible set in problem (5.1.4a) is a convex set. We recall here that (see [3, Lemma 3.5]) δ-γ-concavity of a copula implies δ-α-concavity of the same copula for all $\alpha \geq \gamma$. If $\mathfrak{C}$ fails to be δ-γ-concave, we have nonetheless split the non-convexity induced by the copula from the inherent convexity structure in v. Convexity of function v is ensured by the following result.

Lemma 5.3.6. *([11, Lemma 9]). Let $f : \mathbb{R}^n \to \mathbb{R}$ be a real-valued and convex function. Assume, moreover that, for each $i = 1,\ldots,m$, we can find $\alpha_i \in \mathbb{R}$ such that the functions $\mathfrak{g}_i$ are α_i-concave on level sets $\{x \in \mathbb{R} : \mathfrak{g}_i(x) \geq b_i\}$, and a second set of parameters $\gamma_i \in [1,\infty], \hat{b}_i > 0$ satisfying either one of the conditions i), ii) or iii) in Theorem 5.3.1. Moreover, suppose that $\ell_i \geq \max\{b_i, \hat{b}_i\}$, $i = 1,\ldots,m$. Then v defined in* (5.1.4b) *is a convex function on the set $\{u \in [0,1]^m : u_i \geq F_i(\ell_i),\ i = 1,\ldots,m\}$.*

For any given $u \in \mathcal{D}\mathrm{om}(v)$, suppose that problem (5.1.4b) *admits optimal Lagrange multipliers s_i^u associated with the constraints $F_i(\mathfrak{g}_i(x)) \geq u_i$, $i = 1,\ldots,m$. Then, the vector s^u belongs to the subdifferential of v at the point u, i.e., $s^u \in \partial v(u)$.*

Notice that there is a gap between the convexity results of Theorem 5.3.1 and those of Lemma 5.3.6. The former allows for the parameters $\gamma_i \in (-\infty,\infty]$, whereas the latter requires $\gamma_i \geq 1$. This gap can actually be filled by appropriately redefining v as indicated below:

Corollary 5.3.7. *Let $D(\gamma)$ be as in Definition 5.2.7. Under the assumptions of Lemma 5.3.6 assume the existence of a $\gamma \in (-\infty,\infty)$ such that $\gamma_i \geq \gamma$ for all $i = 1,...,m$ and that $v : D(\gamma) \to \mathbb{R} \cup \{\infty\}$ given in* (5.1.4b) *is (re-)defined as follows:*

$$v(u) = \min_{x\in X} f(x) \quad s.t. \quad F_i(\mathfrak{g}_i(x)) \geq u_i^{\frac{1}{\gamma}},\ i = 1,\ldots,m, \tag{5.3.2a}$$

when $\gamma \neq 0$ *and*

$$v(u) = \min_{x \in X} \; f(x) \;\; s.t. \;\; F_i(\mathfrak{g}_i(x)) \geq \exp(u_i), \; i = 1, \ldots, m, \tag{5.3.2b}$$

when $\gamma = 0$. *Then* v *is a convex function on the set* $\{u \in D(\gamma) \; : u_i^{\frac{1}{\gamma}} \geq F_i(\ell_i), i = 1, ..., m\}$ $(\gamma \neq 0)$ *or* $\{u \in D(\gamma) \; : \exp(u_i) \geq F_i(\ell_i), i = 1, ..., m\}$ $(\gamma \neq 0)$.

Since $u \in [0,1]^m$ can be arbitrary, problem (5.1.4b) may be infeasible. In order to deal with this, we consider the following auxiliary problem:

$$\mathfrak{f}(u) := \left\{ \begin{array}{ll} \min\limits_{x \in X, z \in \mathbb{R}^m_+} & \|z\|_1 \\ \text{s.t.} & F_i(\mathfrak{g}_i(x) + z_i) \geq u_i, \; i = 1, \ldots, m \end{array} \right. \tag{5.3.3}$$

Since $X \neq \emptyset$, problem (5.3.3) is feasible for all $u \in [0,1]^m$. Therefore, $\mathfrak{f}$ is a finite valued function: $\mathfrak{f} : [0,1]^m \to \mathbb{R}_+$. Moreover, $\mathfrak{f}(u) = 0$ if and only if the feasible set of problem (5.1.4b) is nonempty.

Lemma 5.3.8. *([11, Lemma 10]). Under the assumptions of Lemma 5.3.6, function* $\mathfrak{f}$ *given in* (5.3.3) *is convex on* $\{u \in [0,1]^m : u_i \geq F_i(\ell_i), \, i = 1, \ldots, m\}$. *Moreover for any given* $u \in [0,1]^m$, *any optimal Lagrange multiplier* $\mathfrak{s}_i^u$ *associated to the constraints* $F_i(\mathfrak{g}_i(x) + z_i) \geq u_i$, $i = 1, \ldots, m$, *satisfies* $\mathfrak{s}^u \in \partial \mathfrak{f}(u)$. *The linearization*

$$\mathfrak{f}(u) + \langle \mathfrak{s}^u, \tilde{u} - u \rangle \leq 0, \tag{5.3.4}$$

is a feasibility cut for problem (5.1.4a).

5.4 Algorithm: regularized GBD with an interpolation step

We now investigate algorithms for solving problem (5.1.3) through formulation (5.1.4). In view of Theorem 5.3.1 and Lemma 5.3.6 we will assume throughout this section that $v : \mathbb{R}^m \to \mathbb{R} \cup \infty$ in (5.1.4b) is an extended real-valued and convex function, and the feasible set of (5.1.4a) is nonempty and convex for a large enough $p \in (0,1]$. We are, however, assuming that $\mathfrak{C}$ satisfies only generalized concavity assumptions.

If the domain of v contains the feasible set of problem (5.1.4a), then we can solve (5.1.4) (and therefore problem (5.1.3)) by applying the supporting hyperplane algorithm proposed in [204] to the following reformulated problem:

$$\min_{\substack{u \in [0,1]^m \\ \underline{y} \leq y \leq \overline{y}}} \; y \quad \text{s.t.} \quad v(u) \leq y, \;\; \mathfrak{C}(u) \geq p, \;\; u_i \geq F_i(\ell_i), \; i = 1, \ldots, m \, ,$$

where $\underline{y}$ and $\overline{y}$ are *properly chosen* bounds satisfying $\underline{y} \leq v_{\min} \leq \overline{y}$. In the application of interest, the domain of v does not contain, in general, the m-dimensional unit box and therefore feasibility cuts must be added to the above problem. As a result, [204] cannot be directly applied to this reformulation (the algorithm in [204] is suitable for real-valued functions, only). Even endowing the supporting hyperplane algorithm of Veinott with means to handle extended real-valued functions the resulting (new) algorithm is not very appealing: supporting hyperplane method possesses slow convergence and requires many function evaluations for solving the problem. Since function v in (5.5.5) is costly, we would like to employ a method for solving (5.1.4a) that requires as few function evaluations as possible.

In the quest of efficiently solving (5.1.4a), we thus propose a new variant of level bundle methods [121] for nonsmooth optimization problems whose objective function is convex but can assume the value infinity, and wherein the nonlinear constraint mappings satisfy only generalized concavity assumptions. To the best of our knowledge, the proposed optimization method is the first bundle algorithm with global convergence guarantees for such an optimization setting. The full algorithm is described in § 5.4.2, but first we present some key ingredients.

5.4.1 Ingredients: cutting-plane models

Given a sequence of generated trial points $\{\tilde{u}^1, \tilde{u}^2, \ldots, \tilde{u}^k\}$, we will split the index set $\{1, \ldots, k\}$ into two subsets: the optimality index set $\mathcal{O}_k$ gathering indices j such that $v(\tilde{u}^j) < \infty$, and the feasibility index set $\mathcal{F}_k$ gathering indices j such that $\mathfrak{f}(\tilde{u}^j) > 0$. We thus have that $\mathcal{O}_k \cap \mathcal{F}_k = \emptyset$ and $\mathcal{O}_k \cup \mathcal{F}_k = \{1, \ldots, k\}$. Accordingly, we define the cutting-plane models for function v and $\mathfrak{f}$, respectively:

$$\check{v}_k(u) := \max_{j \in \mathcal{O}_k} \left\{ v(\tilde{u}^j) + \langle s^j_{\tilde{u}}, u - \tilde{u}^j \rangle \right\}, \quad \check{\mathfrak{f}}_k(u) := \max_{j \in \mathcal{F}_k} \left\{ \mathfrak{f}(\tilde{u}^j) + \langle \mathfrak{s}^j_{\tilde{u}}, u - \tilde{u}^j \rangle \right\}.$$

Since v and $\mathfrak{f}$ are convex functions, the models $\check{v}_k$ and $\check{\mathfrak{f}}_k$ approximate, respectively, v and $\mathfrak{f}$ from below. As $\mathfrak{C}$ is not concave, we cannot approximate $\mathfrak{C}$ by using first-order linearizations of the type $\mathfrak{C}(u) + \langle \nabla \mathfrak{C}(u), \cdot - u \rangle$ without cutting-off the set

$$\mathfrak{M}(p) = \{u \in [0,1]^m : \mathfrak{C}(u) \geq p\}. \tag{5.4.1}$$

However, since this latter set is convex (e.g., Theorem 5.3.1), we can approximate $\mathfrak{M}(p)$ by using tangent directions, as shown by the following classic result (e.g., [96, Chapter III]).

Lemma 5.4.1. *Let $\mathfrak{C} : [0,1]^m \to [0,1]$ be a continuously differentiable copula such that $\mathfrak{M}(p)$ given in (5.4.1) is a convex set for a given $p \in [0,1]$. Then for any $\tilde{u} \in [0,1]^m$ such that $\mathfrak{C}(\tilde{u}) = p$, the inequality $\langle \nabla \mathfrak{C}(\tilde{u}), u - \tilde{u} \rangle \geq 0$ is a supporting hyperplane for $\mathfrak{M}(p)$. Moreover, if $\nabla \mathfrak{C}(\tilde{u}) \neq 0$ and if $u^{\mathtt{in}}$ is a point in the interior of $\mathfrak{M}(p)$, then, $\langle \nabla \mathfrak{C}(\tilde{u}), u^{\mathtt{in}} - \tilde{u} \rangle > 0$.*

Corollary 5.4.2. *Let $\mathfrak{C} : [0,1]^m \to \mathbb{R}$ be a continuously differentiable Archimedean copula and consider the constraint set $C = \{u \in [0,1]^m : \mathfrak{C}(u) \geq p\}$. Then for any $u \in C$ with $\mathfrak{C}(u) = p$, we have $\nabla \mathfrak{C}(u) \neq 0$.*

Remark 5.4.3. *The existence of a point $u^{\mathtt{in}}$ in the interior of $\mathfrak{M}(p)$, defined in (5.4.1), is not a restrictive assumption. Indeed, for Archimeadean copulæ such a point $u^{\mathtt{in}}$ can be easily computed: let $p \in (0,1)$ and φ_θ (the copula generator function) be given and $\mathbf{1}$ be the vector of ones in $\mathbb{R}^m$. Then $u^{\mathtt{in}}$ defined as $u^{\mathtt{in}} := \mathbf{1}\varphi_\theta^{-1}\big(\frac{\varphi_\theta(\frac{p+1}{2})}{m}\big)$ is readily seen to belong to interior of $\mathfrak{M}(p)$. Note that by taking $u^{\mathtt{in}}$ sufficiently large enough (close to 1), one can ensure that $u_i^{\mathtt{in}} > F_i(\ell_i)$ holds as well.*

Having the cutting-plane information at iteration k, let $v_{\mathtt{low}}^{k+1}$ be the optimal value of the following linear programming problem:

$$v_{\mathtt{low}}^{k+1} := \left\{ \begin{array}{lll} \min\limits_{u} & \check{v}_k(u) & \\ \text{s.t.} & \check{\mathfrak{f}}_k(u) \leq 0 & \\ & \langle \nabla \mathfrak{C}(\tilde{u}^j), u \rangle \geq \langle \nabla \mathfrak{C}(\tilde{u}^j), \tilde{u}^j \rangle, & j \in \big\{ i \leq k : \mathfrak{C}(\tilde{u}^i) = p \big\} \\ & u \in [0,1]^m,\ u_i \geq F_i(\ell_i), & i = 1, \ldots, m\,. \end{array} \right. \tag{5.4.2}$$

In view of Lemma 5.4.1 problem (5.4.2) is an outer-approximation of problem (5.1.4a). As a result, $v_{\mathtt{low}}^{k+1}$ is a lower bound for the optimal value of (5.1.4a): $v_{\mathtt{low}}^{k+1} \leq v_{\min}$ for all $k = 1, 2, \ldots$. Algorithm 5.1 given below generates two sequences of iterates: $\{u^k\}$ and $\{\tilde{u}^k\}$ both satisfying the last two constraints in (5.4.2), where the last sequence satisfies also $\mathfrak{C}(\tilde{u}^k) \geq p$ for all k. Since $\tilde{u}^k$ is a feasible point, $v(\tilde{u}^k)$ is an upper bound for the optimal value of problem (5.1.4a): $v_{\min} \leq v(\tilde{u}^k)$ for all $k = 1, 2, \ldots$. An optimality measure for problem (5.1.4a) is, therefore, the gap

$$\Delta_{k+1} := \min_{j \in \mathcal{O}_k} v(\tilde{u}^j) - v_{\mathtt{low}}^{k+1}. \tag{5.4.3}$$

Indeed, if $\Delta_{k+1} \leq \eta$ for some tolerance $\eta > 0$, we have that

$$\eta \geq \min_{j \in \mathcal{O}_k} v(\tilde{u}^j) - v_{\mathtt{low}}^{k+1} \geq \min_{j \in \mathcal{O}_k} v(\tilde{u}^j) - v_{\min} \geq 0. \tag{5.4.4}$$

(If the index i is defined as $i := \arg\min_{j \in \mathcal{O}_k} v(\tilde{u}^j)$, then $\tilde{u}^i$ is η-optimal for (5.1.4a) since $v(\tilde{u}^i) \leq v_{\min} + \eta$.) We now explain how our algorithm defines the two sequences of iterates $\{u^k\}$ and $\{\tilde{u}^k\}$. In order to

do that, we select a parameter $\kappa \in (0,1)$ and a stability center $\hat{u}^k$ feasible for problem (5.1.4a). The sequence $\{u^k\}$ is generated by solving, at iteration k, the following problem:

$$\begin{cases} \min\limits_{u} & \|u - \hat{u}^k\|^2 \\ \text{s.t.} & \check{v}_k(u) \le v^k_{\texttt{low}} + \kappa\Delta_k \\ & \check{\mathfrak{f}}_k(u) \le 0 \\ & \langle \nabla\mathfrak{C}(\tilde{u}^j), u\rangle \ge \langle \nabla\mathfrak{C}(\tilde{u}^j), \tilde{u}^j\rangle, \quad j \in \{i \le k : \mathfrak{C}(\tilde{u}^i) = p\} \\ & u \in [0,1]^m,\ u_i \ge F_i(\ell_i), \qquad i = 1,\dots,m\,, \end{cases} \tag{5.4.5}$$

We propose to set $\hat{u}^0 := u^{\texttt{in}}$, $k_{\texttt{ref}} := 1$ and update the stability center according to the rule

$$\hat{u}^{k+1} \leftarrow \begin{cases} u^{k+1} & \text{if} \quad \Delta_{k+1} \le (1-\kappa)\Delta_{k_{\texttt{ref}}} \quad \text{(in this case set } k_{\texttt{ref}} \leftarrow k+1\text{)} \\ \hat{u}^k & \text{if} \quad \Delta_{k+1} > (1-\kappa)\Delta_{k_{\texttt{ref}}}\,. \end{cases} \tag{5.4.6}$$

However, other rules for updating the stability center are possible. For instance, we can set $\hat{u}^k = u^k$ for all k or, simply, $\hat{u}^k = u^{\texttt{in}}$ for all k. Let $u^{\texttt{in}}$ as in Lemma 5.4.1 be given, then the sequence of feasible points $\{\tilde{u}^k\}$ is obtained by defining

$$\tilde{u}^k = u^{\texttt{in}} + \lambda^k(u^k - u^{\texttt{in}})\,, \tag{5.4.7}$$

where $\lambda^k \in (0,1]$ is the largest number such that $\mathfrak{C}(\tilde{u}^k) \ge p$. Accordingly, $\tilde{u}^k = u^k$ whenever $\mathfrak{C}(u^k) \ge p$ (i.e., $\lambda^k = 1$), and $\mathfrak{C}(\tilde{u}^k) = p$ if $\mathfrak{C}(u^k) < p$. In this latter case, continuity of $\mathfrak{C}$ ensures that $\tilde{u}^k$ as in (5.4.7) can be computed by employing a bisection procedure on the interval $[0,1]$.

5.4.2 A regularized supporting hyperplane algorithm

We now present our algorithm, which we will refer to as RSHM in the sequel.

Algorithm 5.1. A REGULARIZED SUPPORTING HYPERPLANE ALGORITHM

Step 0 (Initialization) *Let $u^{\texttt{in}}$ be as in Lemma 5.4.1. Set $\mathcal{O}_0 = \mathcal{F}_0 = \emptyset$, $\Delta_1 = \infty$, $\tilde{u}^1 = u^{\texttt{in}}$, $k = 1$ and choose $\eta > 0$ and $\kappa \in (0,1)$.*

Step 1 (Stopping Test) *If $\Delta_k \le \eta$, stop.*

Step 2 (Oracle v) *Try to compute $v(\tilde{u}^k)$ and $s^k_{\tilde{u}} \in \partial v(\tilde{u}^k)$.*
If problem (5.1.4b) is infeasible, set $\mathcal{O}_k \leftarrow \mathcal{O}_{k-1}$ and go to Step 3.
Otherwise, set $\mathcal{O}_k \leftarrow \mathcal{O}_{k-1} \cup \{k\}$, $\mathcal{F}_k \leftarrow \mathcal{F}_{k-1}$ and go to Step 4.

Step 3 (Oracle $\mathfrak{f}$) *Compute $\mathfrak{f}(\tilde{u}^k)$, $\mathfrak{s}^k_{\tilde{u}} \in \partial\mathfrak{f}(\tilde{u}^k)$ and set $\mathcal{F}_k \leftarrow \mathcal{F}_{k-1} \cup \{k\}$.*

Step 4 (Primal Step) *Compute $v^{k+1}_{\texttt{low}}$ by solving the LP problem (5.4.2) and u^{k+1} by solving the master problem (5.4.5).*

Step 5 (Bisection) *If $\mathfrak{C}(u^{k+1}) < p$, compute $\tilde{u}^{k+1}$ as in (5.4.7) and $\nabla\mathfrak{C}(\tilde{u}^{k+1})$. Otherwise, set $\tilde{u}^{k+1} = u^{k+1}$.*

Step 6 (Loop) *Compute Δ_{k+1} as in (5.4.3) and obtain $\hat{u}^{k+1}$ as in (5.4.6) (or another suitable rule). Set $k \leftarrow k+1$ and return to Step 1.*

The case in which either $\mathcal{O}_k$ or $\mathcal{F}_k$ is an empty set deserves comments: (i) if $\mathcal{O}_k = \emptyset$, then (5.4.2) must be interpreted as the problem of finding a point u^{k+1} in the feasible set defined in (5.4.2). In this case, $v^{k+1}_{\texttt{low}}$ should be defined as $-\infty$; (ii) if $\mathcal{F}_k = \emptyset$, then $\check{\mathfrak{f}}_k$ is meaningless, and it should be removed from (5.4.2) and (5.4.5). Since at each iteration the algorithm adds more constraints in problem (5.4.2) (because either $\mathcal{O}_k$ or $\mathcal{F}_k$ is enlarged), we conclude that the sequence $\{v^k_{\texttt{low}}\}_k$ is non-decreasing.

The original level method [121] was designed to solve optimization problems with convex objective and constraint functions (concave functions if we are thinking of constraints of the type $\mathfrak{C}(u) \ge p$). Each iteration of the level method in [121] involves solving two optimization subproblems; first a linear program to compute the *level parameter* $v^k_{\texttt{low}} + \kappa\Delta_k$, and then a projection problem to define a new iterate. This is also the strategy adopted by Algorithm 5.1 that successively solves both subproblems (5.4.2) and (5.4.5) at each iteration of the algorithm. However, solving (5.4.2) to define a lower bound $v^{k+1}_{\texttt{low}}$ is optional. In fact, we can let the lower bound fixed along some iterations until the algorithm identifies that the master

problem (5.4.5) is infeasible. In this case, the lower bound must be increased: the most common rule is to set $v_{\texttt{low}}^{k+1} = v_{\texttt{low}}^k + \kappa\Delta_k$ if (5.4.5) is infeasible, and $v_{\texttt{low}}^{k+1} = v_{\texttt{low}}^k$ otherwise. This rule ensures that, for any $k > 0$, $v_{\texttt{low}}^k$ is a lower bound for the optimal value of (5.1.4a) as long as $v_{\texttt{low}}^0$ satisfies $v_{\texttt{low}}^0 \leq v_{\min}$; see [9, Lemma 2].

5.4.3 Convergence analysis

In order to establish convergence we will need the following auxiliary result.

Lemma 5.4.4. *([11, Lemma 12]). Consider Algorithm 5.1 and assume that $\nabla\mathfrak{C}(\cdot)$ is continuous on $[0,1]^m$ and $0 \neq \nabla\mathfrak{C}(u)$ for all $u \in [0,1]^m$ such that $\mathfrak{C}(u) = p$. Assume that the algorithm generates an infinite sequence of iterates. Let $\tilde{u}$ be a cluster point of the sequence $\{\tilde{u}^k\}$. Then, there exists an index set $K \subseteq \mathbb{N}$ such that $\lim_{k\in K} u^k = \tilde{u} = \lim_{k\in K} \tilde{u}^k$.*

Convergence of Algorithm 5.1 is given in the following theorem.

Theorem 5.4.5. *([11, Lemma 4.1]). Under the assumptions of Lemma 5.3.6, suppose in addition that all subgradients of v generated by the oracle in Step 2 of Algorithm 5.1 are uniformly bounded, $\nabla\mathfrak{C}(\cdot)$ is continuous on $[0,1]^m$ and $0 \neq \nabla\mathfrak{C}(u)$ for all $u \in [0,1]^m$ such that $\mathfrak{C}(u) = p$. Moreover, suppose that $\eta = 0$ and that the algorithm produces infinitely many optimality cuts whose indices are gathered in the index set $\mathcal{O}$. Then, any cluster point $\tilde{u}$ of the sequence $\{\tilde{u}^k\}_{k\in\mathcal{O}}$ generated by Algorithm 5.1 is a solution to problem* (5.1.4a).

The assumption that $\nabla\mathfrak{C}(\cdot)$ is continuous and differs from zero for all $u \in [0,1]^m$ such $\nabla\mathfrak{C}(u) = p$ is satisfied, for instance, for all Archimedean copulæ; see Corollary 5.4.2.

If, after a certain iteration $\bar{k}$, only feasibility cuts are generated, it can be shown that $\lim_k \mathfrak{f}(\tilde{u}^k) = 0$. Hence, any cluster point of $\tilde{u}$ of $\{\tilde{u}^k\}$ belongs to the domain of v.

We finalize this section by mentioning that the given convergence analysis does not require any assumption on the norm $\|\cdot\|$ in (5.4.5). If the objective function $\|u - \tilde{u}^k\|^2$ is replaced with $\|u - \tilde{u}^k\|_1$ or $\|u - \tilde{u}^k\|_\infty$ (i.e., ℓ_1 or ℓ_∞ norms), then the resulting master problem becomes a linear programming problem. This is an interesting feature for dealing with large-scale optimization problems, whose quadratic master solution can be expensive. The convergence speed may be reduced by this change in stabilization. However, this may largely be compensated by a gain in resolution speed because the master problem is less expensive [24, 71, 72].

5.5 Test problems and numerical experiments

We now focus on the numerical solution of problem (5.1.3). The task of choosing a suitable copula that models the dependency among the random variables $\{\omega_1, \ldots, \omega_m\}$ is beyond the scope of this work and will not be discussed here.

The MATLAB sources and test-problem generator (as well as many copulæ) used in Section 5.5.1 below are publicly available at the link `www.oliveira.mat.br/solvers`.

5.5.1 Approximation of a chance-constrained problem with random technology matrix

In this subsection we consider the following special case of problem (5.1.3) where the mappings $\mathfrak{g}_i$ in (5.1.3b) are given by

$$\mathfrak{g}_i(x) = \frac{a_i - \mu_i^\intercal x}{\sqrt{x^\intercal \Sigma_i x}}\,, \tag{5.5.1}$$

with given vectors $\mu_i \in \mathbb{R}^n$ and symmetric positive definite matrices $\Sigma_i \in \mathbb{R}^{n\times n}$ for all $i = 1, \ldots, m$. These mappings satisfy some generalized concavity assumptions, as stated in the following lemma.

Lemma 5.5.1 ((see [39])). *Let $a > 0$ be given, $\mu \in \mathbb{R}^n$ and Σ be an $n \times n$ positive definite matrix. Define the mapping $g : D \to \mathbb{R}_+ \cup \{\infty\}$ as:*

$$\mathfrak{g}(x) = \begin{cases} \frac{a-\mu^\mathsf{T} x}{\sqrt{x^\mathsf{T}\Sigma x}} & \text{if} \quad x \neq 0 \\ \infty & \text{else,} \end{cases} \tag{5.5.2}$$

where D is the set $D = \left\{x \in \mathbb{R}^n \ : \mu^\mathsf{T} x \leq a\right\}$. This mapping is $(-r)$-concave on the set

$$\{x \in D \ : \mathfrak{g}(x) > b(r)\}, \quad \text{with} \ \ b(r) := \frac{r+1}{r-1}(\lambda_{\mathtt{min}})^{-\frac{1}{2}} \|\mu\| \tag{5.5.3}$$

for all $r \in (1, 3]$, where $\lambda_{\mathtt{min}} > 0$ is the smallest eigenvalue of the positive definite matrix Σ. Here we use the convention $\frac{1}{\infty} = 0$.

The above lemma is an improvement on [91].

Given $\mathfrak{g}_i$ in (5.5.1), we assume that each component ω_i of the random vector ω is a standard Gaussian random variable, i.e., $\omega_i \sim \mathcal{N}(0, 1)$ for all $i = 1, \ldots, m$. Sklar's Theorem ensures that there exists a suitable copula $\mathfrak{C}$ such that the probability $\mathbb{P}[\omega \leq \mathfrak{g}(x)]$ can be written as

$$\mathbb{P}[\omega \leq \mathfrak{g}(x)] = \mathbb{P}[\omega_i \leq \mathfrak{g}_i(x),\, i = 1, ..., m] = \mathfrak{C}(\Phi(\mathfrak{g}_1(x)), \ldots, \Phi(\mathfrak{g}_m(x)))\,, \tag{5.5.4}$$

where Φ $(= F_i)$ is the standard Gaussian cumulative distribution function. For instance, if the random variables $\{\omega_1, \ldots, \omega_m\}$ are mutually independent, then $\mathfrak{C}$ above is the product (or independent) copula $\mathfrak{C}(u) = \Pi_{i=1}^m u_i$. In the dependent case, other copulæ must be used instead.

Under these assumptions, problem (5.1.4b) defining function $v(u)$ can be written as:

$$v(u) = \min_{x\in X} \quad f(x) \quad \text{s.t.} \quad \mu_i^\mathsf{T} x + \Phi^{-1}(u_i)\sqrt{x^\mathsf{T}\Sigma_i x} \leq a_i,\ i = 1, \ldots, m\,. \tag{5.5.5}$$

Consequently, if f is linear or quadratic convex, then the above problem is a convex conic optimization problem, and can be efficiently solved by standard methods.

Remark 5.5.2. *Consider the specific instance of problem* (5.1.3) *with constraint mappings* (5.5.1) *and structure* (5.5.4). *Item e) in Theorem 5.3.1 requires that $\ell_i \geq b_i$, $i = 1, \ldots, m$. Since $b_i = b(r)$ defined in* (5.5.3) *is positive, it follows that when $u \in [0, 1]^m$ is feasible for* (5.1.4a), *then $u_i \geq F_i(\ell_i) > F_i(0) = \frac{1}{2}$. In this case $\Phi^{-1}(u_i) > 0$ and the constraints in* (5.5.5) *reads as $\mu_i^\mathsf{T} x - a_i \leq -\Phi^{-1}(u_i)\sqrt{x^\mathsf{T}\Sigma_i x} < 0$ for all $x \neq 0$. Moreover, by assuming that each component a_i is strictly positive the nonlinear constraints in* (5.5.5) *are also satisfied for $x = 0$.*

Motivation. The motivation for this specific form of the mappings $\mathfrak{g}_i$ and ω_i comes from the following special form of constraint mapping

$$\mathbb{P}[\omega_i^\mathsf{T} x \leq a_i, i = 1, ..., m] \geq p, \tag{5.5.6}$$

where each $\omega_i \sim \mathcal{N}(\mu_i, \Sigma_i)$ follows a multi-variate Gaussian distribution. For each $i = 1, ..., m$, the following holds:

$$\mathbb{P}[\omega_i^\mathsf{T} x \leq a_i] = \mathbb{P}\left[\omega_i \leq \frac{a_i - \mu_i^\mathsf{T} x}{\sqrt{x^\mathsf{T}\Sigma_i x}}\right] = \mathbb{P}[\omega_i \leq \mathfrak{g}_i(x)].$$

Let ω be the $m \times n$ matrix stacking the m rows following distribution ω_i. It is readily observed that $\boldsymbol{\eta}(x) := \omega x \in \mathbb{R}^m$ is also a Gaussian random vector with x dependent mean of which the ith component equals $\mu_i^\mathsf{T} x$ and also with x dependent Covariance matrix $\Theta(x)$. For any $i, j = 1, ..., m$, one can show that $\Theta_{ij}(x) = x^\mathsf{T}\Sigma^{ij}x$, where Σ^{ij} is the covariance matrix between rows i and j respectively and $\Sigma^{ii} = \Sigma_i$.

Now (5.5.6) is equivalent to $\mathbb{P}[\tilde{\eta}(x) \leq \mathfrak{g}(x)] \geq p$, where $g : \mathbb{R}^n \to \mathbb{R}^m$ contains as i-th component the mapping $\mathfrak{g}_i$ given in (5.5.1), and $\tilde{\eta}(x)$ is a centered multivariate Gaussian random variable with correlation matrix $\tilde{\Theta}(x)$ having components:

$$\tilde{\Theta}_{ij}(x) = \frac{x^{\mathsf{T}}\Sigma^{ij}x}{\sqrt{x^{\mathsf{T}}\Sigma_i x}\sqrt{x^{\mathsf{T}}\Sigma_j x}}.$$

Therefore, (5.5.4) considerably simplifies the dependency structure in (5.5.6), already by removing the dependency on x in it. Still, dependency can be introduced through the choice of an x-independent Copula and we believe this to be worthwhile. It is clear that (5.5.4) fits the assumptions of this work. Moreover from the viewpoint of (5.5.6), defining $\mathfrak{g}(0) = \infty$ is natural. Indeed, when $a > 0$, it is readily seen in (5.5.6) that $x = 0$ is feasible for all $p \in [0, 1]$. By including $x = 0$ in the definition of g in this manner, it also belongs to the level set (5.5.3) and to the feasible set in (5.5.5), again regardless of p.

5.5.1.1 The problem, solvers, and instances for numerical experiments

From now on we focus on problem (5.1.3) whose objective function f is linear and constraints $\mathfrak{g}_i$, $i = 1, \ldots, m$, are given in (5.5.1).

The problem data $\mu_i \in \mathbb{R}^n$, $\Sigma_i \in \mathbb{R}^{n\times n}$, $a_i \in \mathbb{R}$, $i = 1, \ldots, m$ and $f(x) = c^{\mathsf{T}}x$, with $c \in \mathbb{R}^n$ were generated randomly by using three different seeds for the random number generator. Vector c defining the linear function f was drawn from the sparse Gaussian distribution with mean equal to zero and variance equal to 300, i.e., $\mathcal{N}(0, 300)$. The vectors μ_i, $i = 1, \ldots, m$, were defined by $\mu_i = \frac{1}{10\sqrt{n}}V_i$, where each coordinate of vector $V_i \in \mathbb{R}^n$ was randomly drawn from $\mathcal{N}(0, 1)$. Each matrix Σ_i was generated by using the Matlab function `gallery('randcorr',n)`. Numbers a_i were defined by the following rule, with $\mathbf{1}$ the vector of ones in $\mathbb{R}^m$: $a_i = \mu_i^{\top}\mathbf{1} + \Phi^{-1}(0.55)\sqrt{\mathbf{1}^{\top}\Sigma_i\,\mathbf{1}}$. In our implementation we make sure that the randomly generated data yields $a_i \geq \delta$, for $i = 1, \ldots, m$ and $\delta = 10^{-5}$. The set X in (5.1.3b) was defined by

$$X = \left\{ x \in \mathbb{R}^n : 0 \leq x_j \leq 10,\ j = 1, \ldots, n, \ \text{ and } \ \sum_{j=1}^{n} x_j \geq \delta \right\}.$$

The reason for using $\delta = 10^{-5}$ in the two definitions above is to satisfy the assumption $a > 0$ in Lemma 5.5.1 and to exclude zero from the feasible set of the problem. Indeed note that the mappings (5.5.1) of this application may display degenerate numerical behaviour near zero. Finally, the bounds ℓ_i in (5.1.3b) were defined by $\ell_i \geq t_0^i(\bar{r}_i)$, $i = 1, \ldots, m$, where the function $t_0^i(r)$ given in the Appendix of [11] depends on $b_i(r)$ in (5.5.3) and $\bar{r}_i$ is a(n approximated) solution to the unidimensional optimization problem $\min t_0^i(r)$ s.t. $r \in (1, 3]$. This choice yields that the feasible set (5.1.3b) is convex for all $p \geq p^* := \mathfrak{C}(\Phi(b_1(\bar{r}_1)), \ldots, \Phi(b_m(\bar{r}_1))$.

Different instances of the problem have been obtained by varying m, n and p as follows:

$$n \in \{20,\ 50,\ 100\},\ m \in \{2,\ 5,\ 10,\ 15\} \ \text{ and } \ p \in \{p^*,\ 90\%,\ 97.5\%\}.$$

Furthermore, several Archimedean copulæ are examined: the Clayton, Gumbel, Independent, Joe, Frank and Ali-Mikhail-Haq copulæ, with different values for the parameter θ. In § 5.5.1.4 we employ the Gaussian copula, a non Archimedean copula, for analyzing further the two considered formulations of the problem.

5.5.1.2 Test problem structure

Two formulations are considered for the problem with data described above and function $\mathfrak{g}_i$ given by (5.5.1):

– a monolithic formulation corresponding to problem (5.1.3), i.e.,

$$\begin{cases} \min\limits_{x\in\mathbb{R}^n} & c^\top x & \\ \text{s.t.} & \mathfrak{g}_i(x) \geq \ell_i & i = 1,\ldots,m \\ & \mathfrak{C}(\Phi(\mathfrak{g}_1(x),\ldots,\Phi(\mathfrak{g}_m(x)) \geq p & \\ & \sum_{j=1}^n x_i \geq \delta,\ 0 \leq x_j \leq 10 & j = 1,\ldots,n\,. \end{cases}$$

Since all the functions in the above problem are smooth we can solve this problem by a general purpose solver for nonlinear optimization. We employed `IPOPT` [205] through the OPTI toolbox for Matlab [46].

– The decomposition formulation (5.1.4) with value function $v(u)$ given by (5.5.5). In this approach, problem (5.1.4a) is solved by Algorithm 5.1 and by a supporting hyperplane algorithm based on [204], but with some new features allowing to consider convex and extended real-valued objective function (instead of linear as assumed in [204]). This new algorithm is denoted by the mnemonic `SHM` (supporting hyperplane method). In what follows we refer to Algorithm 5.1 by `RSHM` (regularized supporting hyperplane method).

The Slater points $u^{\mathtt{in}}$ for solvers `SHM` and `RSHM` were defined as in Remark 5.4.3. For the monolithic approach, the OPTI toolbox was setup with the following parameters:

optiset('solver','ipopt','display','iter','maxiter',5e3,'maxfeval',3e4,'maxtime',3.6e3,'tolrfun',5e-5).

Solvers `SHM` and `RSHM` also employed a limit CPU time of one hour. The maximum number of iterations for these two solvers was set to 1000 and relative tolerance for the optimality gap as 5×10^{-5} (the same tolerance set for `IPOPT`).

5.5.1.3 General results

We start this section by analyzing the impact over the optimal value, CPU time and number of iterations by changing the copula in 12 different instances of the problem corresponding to $n = 50$, all four values of m and all three considered values for p. Table 5.1 also reports the obtained threshold p^*. Since the set $\{u \in [0,1]^m : u_i \geq F_i(\ell_i)\}$ in (5.1.4b) (with ℓ_i given at item e) of Theorem 5.3.1) is contained in $\{u \in [0,1]^m : \mathfrak{C}(u) \geq p^*\}$, the instances with $p = p^*$ are easier than those with $p > p^*$. Except for these easy instances, solver `RSHM` uniformly outperformed `SHM` in the considered test-problems. When compared to `IPOPT`, solver `RSHM` outperforms the former one in many instances with copulæ Frank(3) and Ali-M-H(0.9). However `RSHM` is outperformed by `IPOPT` when copula Clayton(1) and $m \in \{10, 15\}$ are considered.

In Figure 5.2 we consider fourteen different copulæ for presenting the solver's performance. The figure reports performance profiles (we refer to [61] for more information) in terms of CPU time and number of copula evaluations (oracle calls) for the three considered solvers. In this case, the higher is the curve the faster is the method (or fewer oracle calls are required).

Table 5.1: Analysis of the three solvers on three different Copulæ.

	Data					# Iterations			CPU Time (s)		
Copula	n	m	p^* (%)	p (%)	f^*	SHM	RSHM	IPOPT	SHM	RSHM	IPOPT
Frank(3)	50	2	91.3	91.3	-563.9	2	6	27	0.1	0.2	1.2
Frank(3)	50	2	91.3	95.0	-487.3	14	11	66	0.3	0.3	2.6
Frank(3)	50	2	91.3	97.5	-423.1	16	9	38	0.4	0.3	1.8
Frank(3)	50	5	80.9	80.9	-456.2	2	7	32	0.3	0.9	2.1
Frank(3)	50	5	80.9	95.0	-341.6	83	38	234	14.6	7.2	13.4
Frank(3)	50	5	80.9	97.5	-305.2	110	46	523	17.0	8.3	29.8
Frank(3)	50	10	70.2	70.2	-558.6	2	3	33	1.1	1.2	3.4
Frank(3)	50	10	70.2	95.0	-404.1	312	112	35	247.2	82.1	3.6
Frank(3)	50	10	70.2	97.5	-365.9	190	96	338	119.6	70.9	31.1
Frank(3)	50	15	60.9	95.0	-659.7	2	2	33	1.4	1.9	4.7
Frank(3)	50	15	60.9	95.0	-448.4	586	205	127	485.7	148.8	16.3
Frank(3)	50	15	60.9	97.5	-410.4	426	200	5000	337.8	161.6	637.1
Clayton(1)	50	2	91.2	91.2	-563.9	2	6	40	0.1	0.2	1.8
Clayton(1)	50	2	91.2	95.0	-485.9	15	11	35	1.2	0.4	1.5
Clayton(1)	50	2	91.2	97.5	-422.6	13	10	36	0.4	0.4	1.6
Clayton(1)	50	5	79.6	79.6	-456.1	2	7	36	0.3	1.1	2.4
Clayton(1)	50	5	79.6	95.0	-340.4	82	40	47	16.4	8.0	4.3
Clayton(1)	50	5	79.6	97.5	-304.7	90	47	32	15.5	9.4	3.0
Clayton(1)	50	10	67.2	67.2	-558.6	2	3	31	0.8	1.4	3.4
Clayton(1)	50	10	67.2	95.0	-402.7	243	118	36	194.7	92.8	3.5
Clayton(1)	50	10	67.2	97.5	-365.0	225	110	31	146.5	76.6	3.2
Clayton(1)	50	15	56.5	56.5	-659.4	2	3	58	1.4	1.8	7.8
Clayton(1)	50	15	56.5	95.0	-446.9	584	193	36	487.0	156.2	6.3
Clayton(1)	50	15	56.5	97.5	-409.5	449	206	65	325.0	159.1	9.1
Ali-M-H(.9)	50	2	91.1	91.1	-563.9	2	6	47	0.1	0.2	1.9
Ali-M-H(.9)	50	2	91.1	95.0	-485.8	16	11	58	0.4	0.3	2.3
Ali-M-H(.9)	50	2	91.1	97.5	-422.6	13	10	87	0.3	0.3	3.5
Ali-M-H(.9)	50	5	79.5	79.5	-456.2	2	7	33	0.3	0.8	2.2
Ali-M-H(.9)	50	5	79.5	95.0	-340.3	83	37	2146	16.1	8.1	121.9
Ali-M-H(.9)	50	5	79.5	97.5	-304.7	94	40	5000	16.1	6.7	284.9
Ali-M-H(.9)	50	10	66.7	66.7	-558.6	2	3	30	0.5	1.5	3.2
Ali-M-H(.9)	50	10	66.7	95.0	-402.6	286	108	88	215.5	73.5	8.3
Ali-M-H(.9)	50	10	66.7	97.5	-365.4	222	119	5000	139.9	80.2	457.7
Ali-M-H(.9)	50	15	55.7	55.7	-446.8	2	3	36	1.4	1.7	5.0
Ali-M-H(.9)	50	15	55.7	95.0	-446.8	576	180	1985	421.0	138.7	249.3
Ali-M-H(.9)	50	15	55.7	97.5	-409.7	452	212	5000	291.7	164.0	635.0

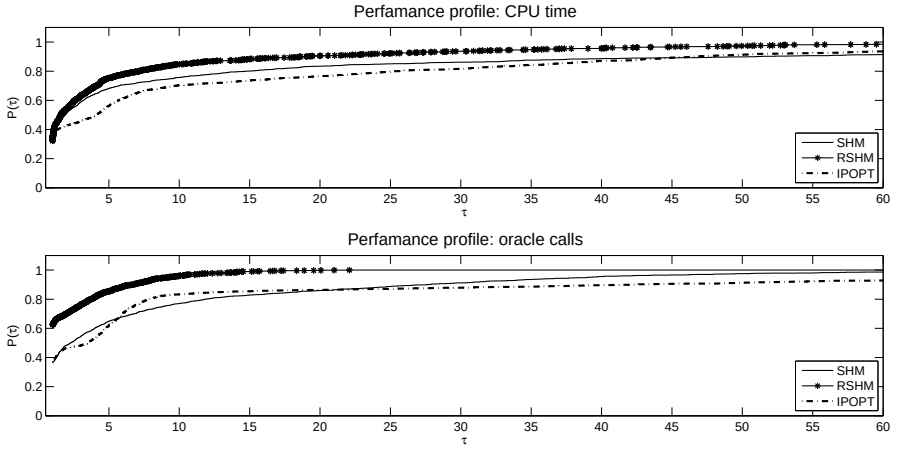


Figure 5.2: Performance profile over 1512 instances. CPU time and number of copula evaluations.

In a total, 1512 instances[1] are considered for each solver. The top subfigure in Figure 5.2 shows that IPOPT was the fastest method in around 38% of all instances, followed by RSHM (32%). However, as

[1]Which correspond to 14 copulæ, 3 values for $p \in \{p^*, 95\%, 97.5\%\}$, 3 different seeds for randomly generating data, 3 values for dimension $n \in \{20, 50, 100\}$ and 4 values for the number of constraints $m \in \{2, 5, 10, 15\}$.

shown in top and bottom subfigures, solver RSHM is more robust that IPOPT in both CPU time and number of copula evaluations: the line corresponding to RSHM approaches the value 1 faster than the other lines. In the bottom subfigure we can see that solver RSHM requires overall less copula evaluations. This is an advantage of RSHM over IPOPT when dealing with copulæ that are difficult to evaluate, such as the Gaussian one analyzed below.

5.5.1.4 Gaussian copula

The Gaussian copula is defined as $\mathfrak{C}^{\texttt{Gauss}}(u) = \Phi_\Sigma(\Phi^{-1}(u_1)), \ldots, \Phi^{-1}(u_m))$, where Φ_Σ is the cumulative distribution function of a multivariate normal distribution with zero mean and covariance matrix Σ and Φ^{-1} the inverse of the one-dimensional (standard) Gaussian distribution function. Therefore, computing $\mathfrak{C}^{\texttt{Gauss}}(u)$ for a given u requires computing numerically a multidimensional integral, which is a difficult task even for moderate dimension m (say, $m = 20$ or more). One would need to resort to the use of Genz' code (e.g., [75]). Since numerical integration is employed, exact values for $\mathfrak{C}$ (and its gradient) can not be expected.

In Table 5.2 we compare the solvers IPOPT and RSHM for 36 instances of the problem defined by the Gaussian copula. The instance size is represented by n-m. Notice that n ranges in the set $\{10, 20, 30, 50\}$ and m in $\{2, 3, 5\}$. Again, three different seeds were employed for the random number generator. Since each

Table 5.2: CPU time average over 3 instances. Values in seconds. Time limit is 3600 (s).

Solver/n-m	10-2	10-3	10-5	20-2	20-3	20-5	30-2	30-3	30-5	50-2	50-3	50-5
RSHM	0.19	0.40	63.25	0.20	0.51	249.68	0.13	0.51	204.45	0.27	0.72	133.51
IPOPT	1.75	4.43	3600*	2.55	5.97	3600*	5.02	8.00	3600*	6.37	15.24	3600*

$\mathfrak{C}$ oracle call is expensive and inexact, the solver IPOPT could not solve the instances with $m = 5$ within one hour. We recall that $m = 5$ involves evaluating a 5 dimensional multi-variate Gaussian distribution function and its derivatives (requiring evaluating $m - 1$ dimensional multivariate Gaussian distribution functions (e.g., [8] and references therein)). Bi- and uni-variate Gaussian distribution functions can be evaluated very efficiently (see [75, Chapter 2]). Consequently the situation $m = 5$, is significantly harder than the situation $m = 3$. For the Gaussian copula the solver RSHM is around 90% faster than IPOPT, since the latter requires more oracle calls. This is an expected fact, since cutting-plane methods are known to be more robust when dealing with noisy functions.

5.5.2 Cascaded-reservoir management

We now investigate a joint-chance-constrained programming problem coming from cascaded reservoir management. For benchmark purposes we consider a real-life configuration of the French hydro valley Isère, described in [7]. The optimization problem can be written as

$$\min_{x \in X} c^\top x \quad \text{s.t.} \quad \mathbb{P}[\omega \leq \mathfrak{g}(x)] \geq p\,, \quad \text{with} \quad \mathfrak{g}(x) = \begin{pmatrix} -a - Ax \\ b + Ax \end{pmatrix}, \tag{5.5.7}$$

$X \subset \mathbb{R}^n$ a bounded polyhedron and $\omega = (-\boldsymbol{\xi}, \boldsymbol{\xi}) \in \mathbb{R}^n$ a random vector. Vectors a, b and matrix A, having appropriate dimensions, are assumed to be fixed. The above problem arises since we wish to make sure that volumes in the reservoirs remain within bounds with high enough probability p. The volumes are impacted by random water inflows ω and turbining strategy. Variable x (belonging to $\mathbb{R}^{566}$) represents the operation planning of power units, and quantity $-c^\top x$ represents the profit yielded by decision x.

In this subsection we assume that each individual random variable ω_i follows a certain Gaussian distribution. We are therefore in the setting of Corollary 5.3.2 and hence $p^* = 0$. By replacing constraint $\mathbb{P}[\omega \leq \mathfrak{g}(x)] \geq p$ by $\mathfrak{C}(\Phi_1(\mathfrak{g}_1(x)), ..., \Phi_m(\mathfrak{g}_m(x))) \geq p$ the slave problem (5.1.4b) and feasibility

problem (5.3.4) become Linear Programming problems. Furthermore, problem (5.5.7) is a nonlinear programming problem with nonlinear and nonconvex constraint, which is solved through `IPOPT`.

We consider two variants of the joint-constrained cascaded reservoir problem (5.5.7): one having $m = 96$ and another with $m = 192$ linear constraints. In each instance, the dimension of the vector x is 566. Table 5.3 reports the number of iterations and CPU time required to solve 28 instances of the problem. Once again, 14 different Copulæ are considered and time limit for all the three solvers `SHM`, `RSHM` and `IPOPT` was set to 1800 s.

As we can see, solver `SHM` failed to solve some instances within the maximum CPU time allowed. Solver `RSHM` was overall faster than `SHM`. Moreover, solver `IPOPT` was outperformed by `RSHM` in most of the instances. The benefits of regularization is evident from Table 5.3. For instance, the number of iterations of `RSHM` is significantly smaller than the number of iterations of solver `SHM` (see the families of Copulæ Gumbel and Joe, for instance).

Table 5.3: Analysis of the three solvers on fourteen different Copulæ. Parameters: $n = 566$, $p = 80\%$ and CPU time limit of 1800 s. The symbol "-" means solver failure.

	Data		# Iterations			CPU Time (s)		
Copula	m	f^*	SHM	RSHM	IPOPT	SHM	RSHM	IPOPT
Clayton(1)	96	-346881.1	201	92	144	22.4	12.7	42.0
Clayton(1)	192	-342927.8	368	195	153	88.7	46.5	48.5
Clayton(3)	96	-346983.3	218	104	182	20.2	9.1	51.6
Clayton(3)	192	-343105.8	375	193	586	67.2	34.9	183.1
Clayton(5)	96	-347083.9	235	118	196	22.7	10.7	54.8
Clayton(5)	192	-343288.3	415	212	516	86.8	42.5	159.9
Indep	96	-346830.3	179	105	190	15.5	9.0	51.9
Indep	192	-342836.3	332	224	163	54.0	46.1	46.4
Gumbel(3)	96	-347489.7	1171	32	418	603.4	2.3	118.0
Gumbel(3)	192	-344285.7	1306	60	818	1800*	7.0	258.8
Gumbel(5)	96	-347592.2	1501	22	323	1213.0	1.4	92.0
Gumbel(5)	192	-344580.2	1267	102	320	1800*	13.8	105.9
Joe(3)	96	-347465.0	1142	30	196	559.8	2.0	55.9
Joe(3)	192	-344228.0	1310	57	58	1800*	6.5	21.9
Joe(5)	96	-347578.0	1501	27	154	1255.7	1.7	44.1
Joe(5)	192	-344539.9	1285	96	242	1800*	12.3	72.8
Frank(-1)	96	-346808.2	189	104	197	16.3	8.8	55.3
Frank(-1)	192	-342802.3	349	211	110	57.5	37.7	34.0
Frank(3)	96	-346925.6	205	112	225	18.4	9.9	64.1
Frank(3)	192	-343006.6	413	182	-	78.4	30.0	-
Frank(5)	96	-347014.1	217	94	461	20.0	7.7	131.0
Frank(5)	192	-343163.3	391	206	50	70.8	38.0	17.9
Ali-M-H(-1)	96	-346780.7	198	82	92	17.2	6.0	26.2
Ali-M-H(-1)	192	-342759.4	331	241	-	50.8	39.9	-
Ali-M-H(0)	96	-346829.2	206	101	195	18.3	8.5	53.7
Ali-M-H(0)	192	-342836.6	327	164	-	51.6	25.0	-
Ali-M-H(.9)	96	-346874.8	208	114	198	18.7	10.1	54.7
Ali-M-H(.9)	192	-342909.1	325	285	-	50.9	69.6	-

In order to analyze the solver performances we considered 84 instances of the problem obtained by fourteen copulæ, two variants of the problem, and three values of $p \in \{80\%, 90\%, 95\%\}$. Figure 5.3 presents the performance profile of the solvers with respect to CPU time and number of copula evaluations (oracle calls). In these numerical experiments, solver `RSHM` was not only the more robust solver but also the fastest one in around 75% of the cases.

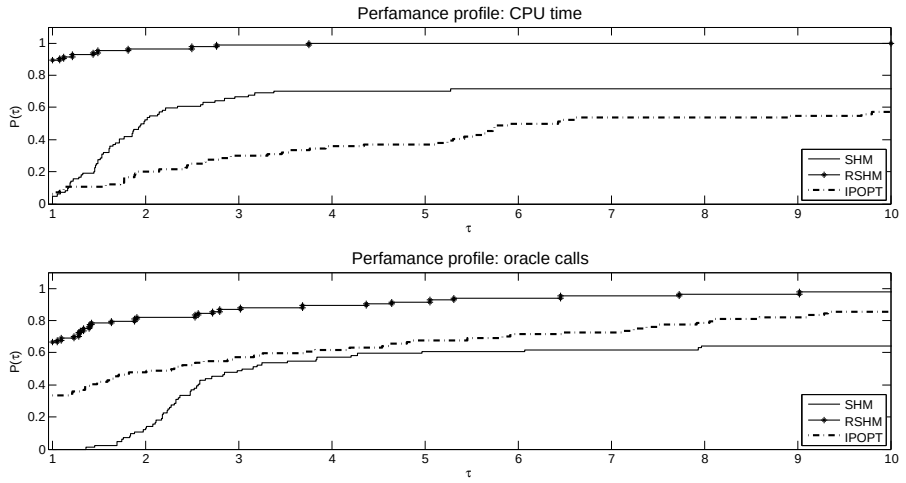


Figure 5.3: Performance profile over 84 instances. CPU time and number of Copula evaluations.

Chapter 6

Probabilistic optimization via approximate p-efficient points and bundle methods

This chapter concerns optimization methods for dealing with separable chance constraint of the form $\mathbb{P}[\omega \leq \mathfrak{g}(x)] \geq p$. Denoting by $\mathcal{V}$ the set of p-efficient points (generalizations of p-quatile) of $\mathbb{P}[\omega \leq \mathfrak{g}(x)]$, the resulting chance-constrained problem can be written in a deterministic fashion

$$\min_{x \in X, v \in \mathcal{V}} f(x) \quad \text{s.t.} \quad \mathfrak{g}(x) \geq v$$

if $\mathcal{V}$ is known. However, the set of p-efficient points is not known is advance and, moreover, each of its elements is difficult to compute. For this reason, efficient optimization methods must come into play. In this chapter, whose results are taken from

> W. van Ackooij, V. Berge, W. de Oliveira and C. Sagastizábal.
> *Probabilistic optimization via approximate p-efficient points and bundle methods*
> **Computers & Operations Research**, *2017, volume 77, pp. 177-193*,

we investigate inexact bundle methods based on p-efficient points to tackle this class of chance-constrained optimization problems.

6.1 Introduction

Probabilistic constraints arise in many real-life problems, for example electricity network expansion, mineral blending, chemical engineering [7, 138, 167, 168]. Typically, these constraints are used when in an ordinary inequality system certain random parameters are identified as critical for the decision making process. We are interested in the so-called *separable* case, in which the random quantities appear only on one side of the constraint set:

$$X(p) := \left\{ x \in \mathbb{R}^n \ : \mathbb{P}[\mathfrak{g}(x) \geq \omega] \geq p \right\}, \tag{6.1.1}$$

where $\mathfrak{g} : \mathbb{R}^n \to \mathbb{R}^m$ is a constraint mapping, $x \in \mathbb{R}^n$ the decision variable, $\omega \in \mathbb{R}^m$ a random vector with probability measure $\mathbb{P}$, and $p \in (0, 1]$ a probability level. Since the mapping $x \mapsto \psi(x) := \mathbb{P}[\mathfrak{g}(x) \geq \omega]$ is nonlinear, writing the constraint in the form $\psi(x) \geq p$ makes (6.1.1) appear as the feasible set of a conventional nonlinear programming problem. However this writing neglects a hidden difficulty: in most

situations explicit values are *not available*. Furthermore, often calculations are inexact, as computing the probability $\psi(x)$ for a given point x typically involves some sort of numerical integration and or (quasi) Monte Carlo methods. Another issue is that the feasible set (6.1.1) sometimes fails to be convex; we refer to [3, 11, 90] for conditions ensuring convexity for sufficiently large probability levels. In order to deal with convex feasible sets regardless of the probability value, throughout we suppose that $\mathfrak{g}(\cdot)$ is concave and the ω-density has generalized concavity properties (e.g., the multivariate Gaussian and Student densities [165]).

An overview of the theory and numerical treatment of probabilistic constraints can be found in [165, 166]. Regarding solution methods, the first approaches [167] were based on cutting planes. More recently, sample average and scenario approximations, [127, 128] and [33, 34] respectively, were employed for linear constraint mappings. The nonlinear programming viewpoint in [9, 10, 31] is likely more suitable for uncertainty with a continuous distribution, as these methods tackle the constraint set (6.1.1) directly by viewing the probability constraint as a nonlinear mapping. Except for [31], convexity of $X(p)$ is assumed and a gradient of the probabilistic constraint needs to be computed.

In this work we follow the lead of [54, 56] and consider solution methods based on the notion of *p-efficient points*, a quantile generalization introduced originally in [163]. The set of p-efficient points is defined as follows:

$$\mathcal{V} := \Big\{ v \in \mathcal{Z} : \text{ no } w \text{ exists in } \mathcal{Z} \text{ such that } w \leq v, w \neq v \Big\} \text{ where } \mathcal{Z} := \{ v \in \mathbb{R}^m \ : \mathbb{P}[\omega \leq v] \geq p \}$$

is the level set of the probability distribution. For the case when $\mathcal{V}$ contains a finite number of elements, early works were concerned with a full enumeration [169]. Notwithstanding, identifying $\mathcal{V}$ can prove difficult, to the extent that in many situations, it is only affordable to compute *one* p-efficient point per iteration.

Methods in this family approximate iteratively the feasible set in (6.1.1) by generating points x for which $\mathfrak{g}(x) \geq v$ for some p-efficient point $v \in \mathcal{V}$. From an optimization perspective, knowing the whole set $\mathcal{V}$ does not really matter: only p-efficient points making active the constraint near a solution are of interest. In a manner similar to column generation in Linear Programming, the process is primal-dual: at the kth iteration, there is a primal pair $(x^k, v^k) \in X(p) \times \mathcal{V}$ and a dual vector u^k related to the constraint $\mathfrak{g}(x) \geq v$. The p-efficient methodology is suitable for both discrete and continuous random vectors, as long as constraints are separable.

The well-known supporting hyperplane methods [134, 167] generate p-efficient points when computing a subgradient to define a direction. The recent bundle approaches in [9, 10] work directly in the nonlinear programming framework without generating p-efficient points, but rather defining linearizations that may support the set $\mathcal{Z}$, for a level below p. By contrast, the approximate p-efficient points used in this work (see Section 6.3), belong to a set $\mathcal{Z}$ for a level *above* p. Another difference with the nonlinear programming approach is that no gradient of the probability constraint needs to be computed with the approach suggested here. Furthermore convergence to an optimal solution of the "direct" bundle methods [9, 10] relies on convexity of the underlying set of feasible solutions. In contrast, in principle the p-efficient point methods do not necessarily require a convexity hypothesis and by moving to the Lagrangian dual (as done in this work too), potential non-linearity of $\mathfrak{g}$ is separated from dealing with the multivariate distribution function of ω. In contrast in the non-linear programming based methods (e.g., [9, 10]) one would need to deal with $\mathfrak{g}$ inside the multivariate distribution function directly. A gradient could be computed through the use of a chain rule or result from a more general formula ([6, 200]). Then a mechanism needs to be designed involving the precision with which the gradient is computed and precision control of the overall algorithm along the lines of [10, Section 5] or [86]. An advantage of the non-linear programming approach is that, in principle, it works for any coupling of the decision and random vector, whereas p-efficient point methods rely on the separability assumption.

In our understanding, the combination of regularization techniques with the iterative primal-dual generation of p-efficient points is behind the excellent results reported in [56, 57], confirmed in our numerical experiments in Section 6.6 below. Our study reveals several interesting relations between those works and bundle methods. Thanks to this connection we develop a general framework for approaches based

on p-efficient points, using as unifying view the proximal bundle theory [149]. This recent bundle variant, dealing with *on-demand accuracy*, was designed to use information from an *oracle* whose calculations have varying precision, following the directives of the bundle solver. In the p-efficient point setting, this amounts to computing such points inexactly, with variable accuracy. This technique makes it possible to solve the problem *exactly* by starting the algorithmic procedure with coarse estimations, increasing exactness of the oracle calculations as the iterations progress. On-demand algorithms keep the convergence properties of classical bundle methods and, as shown by our numerical experiments, can provide very significant gains in CPU time without losing accuracy in the solution.

Our approach, which can be applied to both discrete and continuous random variables, represents a contribution in three fronts. First, by extending the theory in [149] to the primal-dual setting, we reveal in Lem. 6.4.4 and Thm.6.4.5 the impact of inexactness in primal terms. Second, we design new on-demand accuracy approaches, including PAL5 in Section 6.6, which performed the best in our numerical experiments. Third, thanks to the unifying view, we show convergence for a generalization of both the Regularized Dual Decomposition and the Progressive Augmented Lagrangian algorithm [56, 57]. The extension incorporates varying regularization/augmentation parameters and the so-called bundle selection or compression mechanism. This additional flexibility proves fundamental for numerical experiments; we refer to Sections 6.4 and 6.6 for details.

This work is organized as follows. Section 6.2 revises concepts and methods in probabilistic optimization. For both discrete and continuous distributions, Section 6.3 discusses oracles that compute, in an on-demand mode, approximate p-efficient points. Section 6.4 describes the basics of on-demand-accuracy bundle methods in the considered setting and gives primal and dual convergence results. The relation with the Regularized Dual Decomposition [57] and the Progressive Augmented Lagrangian method [56] is explained in Section 6.5. This section also contains the new algorithms introduced in this work. Section 6.6 studies the performance of fifteen different solvers on several instances of cash-matching, cascaded reservoir management, and probabilistic transportation problems. A thorough analysis, reporting CPU times and quality of the solution both in terms of optimality and feasibility, gives a clear panorama of the merits of the different methods in the benchmark.

Our notation is standard. The inner product and induced norm are $\langle \cdot, \cdot \rangle$ and $\|\cdot\|$. For a convex function f, a point $u \in \mathbb{R}^m$ and $\eta \geq 0$, the exact and approximate convex analysis subdifferentials are denoted by $\partial f(u)$ and $\partial_\eta f(u)$. For a concave function φ we consider the corresponding objects of its negative, i.e., $\partial(-\varphi)(u)$ and $\partial_\eta(-\varphi)(u)$. The normal cone of the nonnegative orthant in $\mathbb{R}^m$ at u is $N_{\mathbb{R}^m_+}(u) = \{p \in \mathbb{R}^m : p \leq 0 \text{ and } \langle p, u \rangle = 0\}$.

6.2 The probabilistic optimization problem

We recall background material relative to probabilistic constraints from [54, 55, 56].

6.2.1 Blanket conditions

We suppose the multivariate random variable $\omega \in \mathbb{R}^m$ has an α-concave distribution, so that the following relations, from [54, Thms. 4.42, 4.60, 4.63, and Lems.4.57 and 4.59], hold:

$\mathcal{Z} = \bigcup \{v + \mathbb{R}^m_+ : v \in \mathcal{V}\}$ is convex, nonempty and closed, $\operatorname{conv} \mathcal{V} \subset \mathcal{Z}$, and $\mathcal{V}$ is bounded from below.

This assumption is convenient in so much as that it allows us to simplify the presentation on the impact of the recovered solution in terms of primal optimality. Otherwise the assertions concerning optimality of the primal found solution would simply need to be interpreted in the view of a specific convexified primal problem which can be made explicit with the tools developed in [122]. We also care to emphasize the difference between the notion of α-concave distribution functions of continuously distributed random variables and discrete random variables. Development of α-concavity theory for discrete distributions has started with [58] and is much less developed that for continuous distributions dating back to the

pioneering works of Prékopa (e.g., [164]). In the former case results such as [54, Theorem 4.65] can be employed to arrive at appropriate assertions concerning $\mathcal{Z}$. Second we highlight that it is quite common to rely on sampling to move from a continuous to a discrete distribution (e.g., [57, Examples 1 & 2]), the relation of which with respect to the original distribution is also well studied (e.g., [128]).

Given a convex function $f : \mathbb{R}^n \to \mathbb{R}$, a nonempty and simple convex compact set $X \subset \mathbb{R}^n$ is (for example a polyhedron), and a concave mapping $\mathfrak{g} : \mathbb{R}^n \to \mathbb{R}^m$, we suppose that our problem of interest

$$\min_{x \in X} f(x) \quad \text{s.t.} \quad \mathbb{P}[\mathfrak{g}(x) \geq \omega] \geq p\,, \tag{6.2.1}$$

has a nonempty solution set and, hence, a finite optimal value, $f^{\min}$. In view of the relations above for $\mathcal{Z}$, the problem below, obtained by variable splitting, is convex:

$$\min_{(x,v) \in \mathbb{R}^{n \times m}} f(x) \quad \text{s.t.} \quad \mathfrak{g}(x) \geq v\,,\ x \in X \ \text{ and } \ v \in \mathcal{V}\,. \tag{6.2.2}$$

For this problem to be well-defined, nonemptiness of the feasible set is usually ensured by a Slater condition. If boundedness of the multipliers in (6.2.2) is a concern, the stronger (Mangasarian-Fromowitz-like) constraint qualification below can be used,

$$\exists (x^s, v^s) \in X \times \mathcal{Z} : \mathfrak{g}(x^s) > v^s \text{ and, in } X \times \mathcal{Z}, \ \left\{ \begin{array}{l} \text{affine equality constraints are linearly independent} \\ \text{inequality constraints are satisfied strictly by } (x^s, v^s)\,. \end{array} \right. \tag{6.2.3}$$

For the problem on dual variables

$$\max_{u \in \mathbb{R}^m_+} \varphi(u) \ \text{ with } \ \varphi(u) := h(u) + d(u), \ \text{ where } \ \left\{ \begin{array}{l} h(u) := \min\limits_{x \in X} \{f(x) - \langle u, \mathfrak{g}(x) \rangle\} \\ d(u) := \min\limits_{v \in \mathcal{V}} \langle u, v \rangle\,, \end{array} \right. \tag{6.2.4}$$

defining d as a minimum is possible ($u \geq 0$ and $\mathcal{V}$ has a lower bound). Also, by the disjunctive expression for $\mathcal{Z}$,

$$d(u) = \min_{v \in \mathcal{V}} \langle u, v \rangle = \min_{v \in \operatorname{conv} \mathcal{V}} \langle u, v \rangle = \min_{v \in \mathcal{Z}} \langle u, v \rangle \quad \text{ for all } u \in \mathbb{R}^m_+\,, \tag{6.2.5}$$

because the minimand is linear. As a result, φ in (6.2.4) coincides with the dual function of problem (6.2.2). By weak duality using the point (x^s, v^s) from (6.2.3), the dual function is bounded above ($\varphi(u) \leq f^{\min}$ for all $u \in \mathbb{R}^m_+$), thus justifying the use of a maximum instead of a supremum in (6.2.4). Furthermore, since there is no duality gap between (6.2.2) and (6.2.4), by [56, Cor. 1] both (6.2.1) and (6.2.2) have (6.2.4) as dual problem and the respective optimal values coincide.

The more general setting in [56], allowing for unbounded sets X, requires additional conditions, such as solvability of the h-problems in (6.2.4), which are automatic in our case, because X is bounded.

6.2.2 Primal and dual views

In (6.2.2), the difficult set $\mathcal{Z}$ is not available explicitly. Like in a Dantzig-Wolfe approximation, solution approaches adopting a primal view keep track of past information, $v^1, v^2, \ldots, v^k$, and consider the unit simplex associated with the index set $J_k = \{1, \ldots, k\}$:

$$\Delta^{|J_k|} := \left\{ \alpha \in \mathbb{R}^{|J_k|}_+ : \ \text{with } \sum_j \alpha_j = 1 \right\}.$$

A new p-efficient point v^{k+1} is generated by using the optimal multiplier of the inequality constraint in

$$\min f(x) \quad \text{s.t.} \quad \mathfrak{g}(x) \geq v\,,\ x \in X \ \text{ and } \ v \in V_k := \left\{ \sum_{j \in J_k} \alpha_j v^j : \alpha \in \Delta^{|J_k|} \right\}. \tag{6.2.6}$$

We refer to [55, 58] for details; see also [117] and [169].

Dual methods such as [56] relax the constraint $\mathfrak{g}(x) \geq v$ in (6.2.2) and maximize the dual function $h + d$ from (6.2.4). Computing the h-term amounts to solving a convex problem (X is a simple set), the difficulty lies in the calculation of $d(u)$ (involving the set $\mathcal{V}$ or its convex hull, by (6.2.5)). Even for distributions with finite support, a challenging mixed-integer linear programming (MILP) problem needs to be solved; see [116, 129]. This confirms the relevance of designing algorithms that can deal with approximate p-efficient points.

As shown in [56, Sec. 3], when both the h- and d-problems in (6.2.4), (6.2.5) are solved exactly at a given point $u = u^j$,

$$\begin{aligned} s_h^j &:= -\mathfrak{g}(x^j) \text{ satisfies } -s_h^j \in \partial(-h)(u^j), \text{ for } x^j \in X \text{ minimizing } f(x) - \langle u^j, \mathfrak{g}(x)\rangle \quad (6.2.7)\\ s_d^j &:= v^j \text{ satisfies } -s_d^j \in \partial(-d)(u^j), \text{ for } v^j \in \operatorname{conv}\mathcal{V} \text{ minimizing } \langle u^j, v\rangle, \quad (6.2.8)\\ s^j &:= s_h^j + s_d^j \text{ satisfies } -s^j \in \partial(-\varphi)(u^j)\,. \end{aligned}$$

Recall that, without loss of generality, the minimizer v^j in (6.2.8) can be taken in $\mathcal{V}$, $\operatorname{conv}\mathcal{V}$, or $\mathcal{Z}$, by (6.2.5).

By assuming that the support set of ω is finite (and the random vector dimension is not too large), an alternative procedure to efficiently solve (6.2.1) can be obtained by using the combinatorial patterns in [114, 115], explained in the sequel.

6.2.3 Combinatorial pattern: an alternative to p-efficient-based approaches

Suppose the support set of ω is finite. The main idea in [114, 115] is to pre-specify a set of cut-points $c_{1,j}, ..., c_{n_j,j}$ for each component $j = 1, ..., m$ of the random vector ω. The first cut-point is $c_{1,j} := F_j^{-1}(p)$ for F_j the j^{th} marginal distribution function of ω_j. The other cut-points $c_{\ell,j}$ belong to the set $Z_j := \{\omega_j^k : F_j(\omega_j^k) \geq p\}$, whose cardinality is n_j. The random realizations ω^i are binarized and the jth component represented as $(\beta_{1,j}^i, ..., \beta_{n_j,j}^i)$, where for a given $j = 1, ..., m$ and $\ell = 1, ..., n_j$, $\beta_{\ell,j}^i = 1$ if $\omega_j^i \geq c_{\ell,j}$ and zero otherwise. For $B = \sum_{j=1}^m n_j$, problem (6.2.1) can be written as

$$\begin{cases} \min\limits_{(x,r)\in X\times\{0,1\}^B} & f(x) & & \\ \text{s.t.} & \mathfrak{g}_j(x) & \geq \sum_{\ell=1}^{n_j} c_{\ell,j} r_{\ell,j} & j = 1, ..., m \\ & \sum_{j=1}^m \sum_{\ell=1}^{n_j} \beta_{\ell,j}^i\, r_{\ell,j} & \leq m - 1 & \forall\, i \in \bar{\Omega}_B^- \\ & \sum_{\ell=1}^{n_j} r_{\ell,j} & = 1, & j = 1, ..., m, \end{cases} \quad (6.2.9)$$

where $\bar{\Omega}_B^-$ contains all p-insufficient realizations with $\beta_{1j} = 1$ for $j = 1, ..., m$. The theory developed in [114, 115] ensures that formulation (6.2.9) is an exact variant of (6.2.1) if, for $j = 1, \ldots, m$, the set of cut-points $c_{1,j}, \ldots, c_{n_j,j}$ includes all the different elements of Z_j. This binarization process has been extended to optimization problems with stochastic quadratic inequalities in [118].

An important advantage of the binarization process is that the set of binary vectors allows for a linear description of the set of feasible realizations; e.g. (6.2.9). Strengths of this strategy are the following, [118]: the approach can handle any type of dependency between random variables and any type (linear, quadratic, etc.) of stochastic inequalities; the size of the reformulated problem (6.2.9), in particular its number of binary variables, do not grow linearly with the number of scenarios used to represent uncertainty, but with the number of cut points used in the binarization process. As a result, for the binarization approach to be successful the number n_j of cut-points should not be too large: problem (6.2.9) has $m + \sum_{j=1}^m n_j$ variables, and can have as many as $\Pi_{j=1}^m n_j$ constraints. Hence, the assumption that (the low dimensional random vector) ω has finite support becomes crucial for this approach.

Since we do not assume neither finite support nor low-dimensional random vector ω, p-efficient approaches become more appropriate under the hypothesis of separable chance constraints. We therefore stick with p-efficient-based strategies and refer interested readers to the outstanding theory in [114] and the excellent results reported in [115] and [118].

6.3 Approximate p-efficient points

Most of the dual approaches identify a new p-efficient point by solving the d-problem (6.2.5) at the current dual iterate, u^j. Since this calculation involves knowing the set $\mathcal{V}$, this is a computationally heavy task. To alleviate the oracle calculations, we consider that the information is delivered with an inaccuracy η_{u^j}, as follows:

$$\left\{\begin{array}{lcll} h_u^j & = & f(x^j) - \langle u^j, \mathfrak{g}(x^j)\rangle & \text{and} \quad s_h^j = -\mathfrak{g}(x^j) \text{ for } x^j \in X, \\ d_u^j & = & \langle u^j, v^j\rangle & \text{and} \quad s_d^j = v^j \text{ for } v^j \in \operatorname{conv}\mathcal{V}, \\ \varphi_{u^j} & = & h_u^j + d_u^j & \text{and} \quad s^j = s_h^j + s_d^j \text{ are such that}, \\ \varphi_{u^j} & & \text{satisfies } \varphi_{u^j} \in [\varphi(u^j), \varphi(u^j) + \eta_{u^j}] & \text{and} \quad -s^j \in \partial_{\eta_{u^j}}(-\varphi)(u^j), \ \text{ with } \ \eta_{u^j} \geq 0\,. \end{array}\right. \tag{6.3.1}$$

We present two variants for the inexact d-evaluation, obtained by generating elements in the level set $\mathcal{Z}$, (larger than $\mathcal{V}$, but convex, recall Section 6.2.1). We refer to these elements as *approximate p-efficient points*.

6.3.1 Discrete distributions

Suppose the random vector $\omega \in \mathbb{R}^m$ has finitely many realizations $\omega^1, \omega^2, \ldots, \omega^N$ with associated probabilities $\pi_1, \pi_2, \ldots, \pi_N$. As shown in [129], problem (6.2.5) amounts to the following MILP:

$$d(u) = \left\{\begin{array}{lll} \min\limits_{(v,z)\in\mathbb{R}^{m\times N}} & \langle u, v\rangle & \\ \text{s.t.} & \omega^i(1-z_i) & \leq v - bz_i, \ i = 1, \ldots, N\,, \\ & \sum_{i=1}^N \pi_i z_i & \leq 1 - p\,, \\ & z_i \in \{0,1\}, & i = 1, \ldots, N\,, \end{array}\right. \tag{6.3.2}$$

where $b \in \mathbb{R}^m$ has components $b_j := \min_{1\leq i\leq N} \omega_j^i$, $j = 1, \ldots, m$, and $u \in \mathbb{R}^m_+$ is given.

For any feasible pair $(\bar{v}, \bar{z})$ in (6.3.2), the $\bar{v}$-component is an approximate p-efficient point: problem (6.3.2) gives the "best" one. For large N, the computational effort of solving (6.3.2) can be prohibitive, and several authors have considered "cheaper" reformulations, starting with [129]. In [116], the N scenarios are preprocessed to reduce the number of random realizations, while [154] presents an alternative based on certain quantization process.

We combine these approaches with a fast heuristic, based on a reformulation with only N 0-1 variables, instead of the $m \times N$ mixed-integer variables in problem (6.3.2). Specifically, for a feasible pair $(v, z) \in \mathbb{R}^m \times \{0,1\}^N$ in (6.3.2),

$$\sum_{i\in I_0(z)}^{N} \pi_i \geq p \ \text{ and } \ \omega^i \leq v \text{ for all } \ i \in I_0(z) := \{1 \leq l \leq N : \ z_l = 0\}\,.$$

As a result, problem (6.3.2) is equivalent to solving the combinatorial problem

$$d(u) = \min_{z\in\{0,1\}^N} d_u(z) \text{ s.t. } \ \sum_{i=1}^N \pi_i z_i \leq 1 - p\,, \quad \text{with} \quad d_u(z) := \min_{v\in\mathbb{R}^m} \langle u, v\rangle \text{ s.t. } \ \omega^i \leq v \ \text{ for all } \ i \in I_0(z)\,. \tag{6.3.3}$$

Even though the function d_u is neither convex nor continuous on $z \in \{0,1\}^N$, its computation is extremely easy, as the minimum in (6.3.3) is attained at the point

$$\tilde{v} \in \mathbb{R}^m \ \text{ such that } \ \tilde{v}_j := \max_{i\in I_0(z)} \omega_j^i \ \text{ for all } \ j = 1, \ldots, m\,, \quad \text{with optimal value} \quad d_u(z) = \sum_{j=1}^m u_j \left[\max_{i\in I_0(z)} \omega_j^i\right]. \tag{6.3.4}$$

In order to tackle (6.3.4) numerically, we employ two heuristics given below.

Heuristic h1 (Incremental Selection. Input: $u \in \mathbb{R}^m_+$, $p > 0$, as well as ω^i and π_i for all $i = 1, \ldots, N$)

Step 1. Take $z = 0 \in \mathbb{R}^N$ and $J_+ = \{j : u_j > 0\}$.

Step 2. Define $\tilde{v}$ as in (6.3.4) and $A = \{i : z_i = 0\} \cap \{i : \omega^i_j = \tilde{v}_j \text{ for some } j \in J_+\} \cap \{1, \ldots, N\}$.

Step 3. For all $i \in A$, define new trial points $\tilde{z}^i = z$ with $z^i_i = 1$ ($\tilde{z}^i$ differs from z only by its i^{th} component).

Step 4. If all trial points $\tilde{z}^i$ are infeasible for (6.3.3), stop. Return $\tilde{v}$ and $d_u(z) = \langle u, \tilde{v}\rangle$.

Step 5. Evaluate d_u from (6.3.4) at all new trial points $\tilde{z}^i$ feasible for (6.3.3). Set $j \in \operatorname{argmin}_{i\in A} d_u(\tilde{z}^i)$, $z = z^j$ and go back to Step 2.

Heuristic h2 (On-Demand Accuracy, input as in Heuristic h1)

Procedure: Stop the MILP solver for (6.3.2) as soon as it finds a feasible point.

As each cycle between Steps 2 and 5 switches a single component of z ($z_i = 0$ becomes $z_i = 1$), Heuristic h1 terminates after finitely many cycles. As for Heuristic h2, since the objective function in (6.3.2) is linear, any feasible point realizes the η-subgradient inequality in (6.3.1).

Example 6.3.1. *In order to illustrate Heuristic h1 consider the following bi-dimensional example with* $p = 0.70$ *and* $u = (1, 1.2)^\top$ *in* (6.3.2)*, and ten equiprobable scenarios given in Table 6.1. To solve* (6.3.3)

i	1	2	3	4	5	6	7	8	9	10
ω^i_1	-1	1	1	4	5	4	6	4	5	2
ω^i_2	3	9	4	9	10	7	8	4	3	0
π_i	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

Table 6.1: Ten scenarios $\omega^i \in \mathbb{R}^2$

we properly select, out of the ten scenarios, three indices i_1, i_2, i_3 *so that* $z_{i_1} = z_{i_2} = z_{i_3} = 1$ *and the other components are all null. Indeed, by* (6.3.4)*, if* $z_i = 0$ *for all* i *then* $\tilde{v} = (6, 10)^\top$ *as shown by the dashed lines in Figure 6.1. Lower values for* $\tilde{v}_j$, $j = 1, 2$ *are better to minimize* $\langle u, \tilde{v}\rangle = \tilde{v}_1 + 1.2\,\tilde{v}_2$ *for* $u \geq 0$*. We should therefore discard three scenarios yielding the minimum value of* $\tilde{v}_1 + 1.2\,\tilde{v}_2$*. Out of the 120 possible combinations, h1 checks only those preventing* $\tilde{v}_1$ *and* $\tilde{v}_2$ *from being small. For instance, scenario* ω^5 *yields* $\tilde{v}_2 = 10$ *and scenario* ω^7 *implies* $\tilde{v}_1 = 6$*, as shown in Figure 6.1. This results in* $A = \{5, 7\}$ *at the first iteration of Heuristic h1. Notice that if* ω^5 *is discarded, setting* $z_5 = 1$ *and* $z_i = 0$*,*

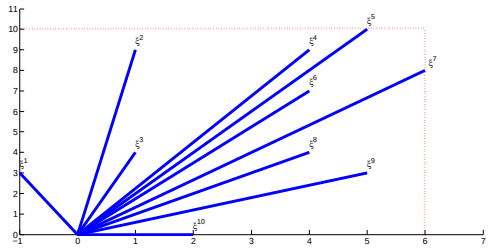
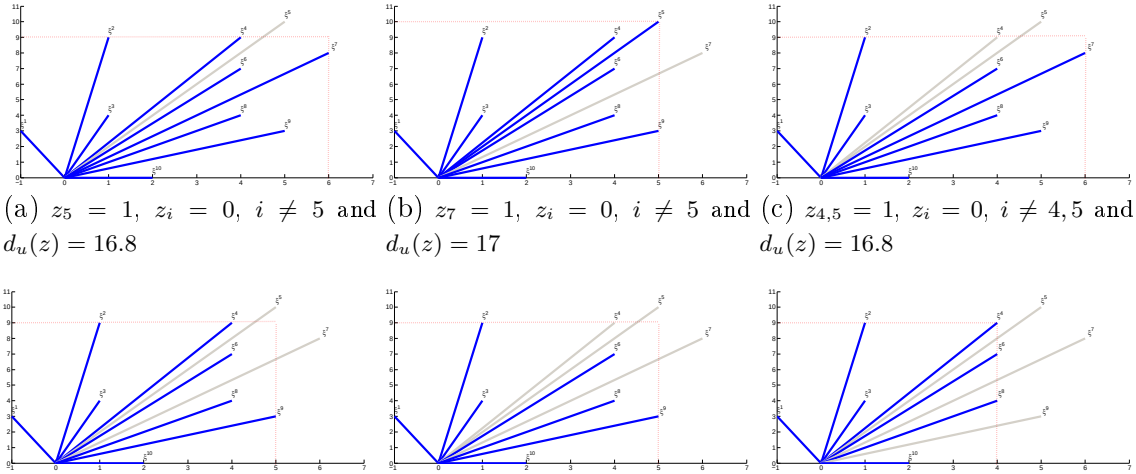


Figure 6.1: Ten scenarios: $z_i = 0$, $i = 1, \ldots 10$ and $d_u(z) = 18$

for all $i \neq 5$ *gives* $\tilde{v}_2 = 9$*, so* $d_u(z) = \langle u, \tilde{v}\rangle = 16.8$*. On the other hand, if* ω^7 *is discarded,* $\tilde{v}_1 = 5$



(a) $z_5 = 1$, $z_i = 0$, $i \neq 5$ and $d_u(z) = 16.8$ (b) $z_7 = 1$, $z_i = 0$, $i \neq 5$ and $d_u(z) = 17$ (c) $z_{4,5} = 1$, $z_i = 0$, $i \neq 4, 5$ and $d_u(z) = 16.8$

(d) $z_{5,7} = 1$, $z_i = 0$, $i \neq 5, 7$ and $d_u(z) = 15.8$ (e) $z_{4,5,7} = 1$, $z_i = 0$, $i \neq 4, 5, 7$ and $d_u(z) = 15.8$ (f) $z_{5,7,9} = 1$, $z_i = 0$, $i \neq 5, 7, 9$ and $d_u(z) = 14.8$

Figure 6.2: Steps of heuristic h1 in the case of Example 6.3.1

and therefore $d_u(z) = 17$. *This procedure is illustrated in Figures 6.2(a) and (b). As* $d_u(z)$ *is lower when scenario* ω^5 *is discarded, Heuristic h1 removes* ω^5 *from the set of scenarios (fixing* $z_5 = 1$*) and continues the procedure. Now scenarios* ω^2 *and* ω^4 *(resp.* ω^7*) prevent* v_2 *(resp.* $\tilde{v}_1$*) from decreasing; see Figures 6.2(c) and (d). So* $A = \{2, 4, 7\}$ *at the second iteration of Heuristic h1. Discarding scenario* ω^7 *reduces* $\tilde{v}_1$ *to 5, and* $\langle u, \tilde{v} \rangle = 15.8$. *To reduce* $\tilde{v}_2$ *it suffices to check either* ω^2 *or* ω^4 *(we just checked* ω^4*); see Figure 6.2 (c). Without scenarios* ω^5 *and* ω^7, v_2 *(resp.* $\tilde{v}_1$*) cannot decrease because of* ω^2 *and* ω^4 *(resp.* ω^9*); see Figures 6.2(e) and (f): at the third iteration,* $A = \{2, 4, 9\}$. *Finally, by checking* ω^4 *and* ω^9 *we conclude that* $\langle u, \tilde{v} \rangle$ *is lower when scenario* ω^9 *is discarded:* $\tilde{v} = (4, 9)^\top$ *is an approximate p-efficient point for the set of scenarios in Table 6.1; see Figure 6.2(f). Heuristic h1 is exact for this example:* $\tilde{v} = (4, 9)^\top$ *is indeed a p-efficient point.*

For the example above, h1 checks only 6 combinations, out of the 120 possibilities. If $\delta := \text{int}(N(1-p))$, the largest integer smaller than $N(1-p)$, and $u \in \mathbb{R}^m$ has $\bar{m}$ nonzero components, then h1 checks at most $\delta \bar{m}$ scenario combinations (in the example $\delta = 3$ and $\bar{m} = 2$). So when p is large and $\delta = 1$, h1 finds the exact solution, because all the combinations decreasing the components of $\tilde{v}$ will be checked. As stated, h1 performs in a forward manner, by successively taking a zero component of vector z and setting it to one. A backward strategy for h1 would initialize $z \equiv 1 \in \mathbb{R}^N$, iteratively setting to zero some components, until $\sum_{i=1}^N \pi_i z_i \leq 1 - p$. This backward strategy is appropriate when $p < 0.5$, while h1 is more suitable when $p > 0.5$.

The idea of solving (6.3.2) inexactly actually lies in obtaining an estimate of d quickly in order to jauge if the current dual vector u is of sufficient interest. If such is the case, a more accurate computation of (6.3.2) can be triggered, taking advantage of "hot-starting", if possible. When the current dual vector is not sufficiently interesting, evaluating d more accurately is useless. These assertions will be made precise in Section 6.4 below. Before we exemplify the combinatorial pattern approach of [114] (and commented in (6.2.3)) by using our toy problem.

Example 6.3.2. *For the toy problem in Example 6.3.1 (with* $x = v$ *and* $c(x) = \langle u, v \rangle$*), cut-points are the elements of the sets* $Z_1 = \{4, 5, 6\}$ *and* $Z_2 = \{8, 9, 10\}$. *The Cartesian product of these two sets generates six new scenarios:* $\omega^{11} = (4, 8)^\top$, $\omega^{12} = (4, 10)^\top$, $\omega^{13} = (5, 8)^\top$, $\omega^{14} = (5, 9)^\top$, $\omega^{15} = (6, 9)^\top$ *and* $\omega^{16} = (6, 10)^\top$. *Among these new scenarios, only* ω^{11} *and* ω^{13} *do not satisfy* $\mathbb{P}[\omega \leq \omega^k] \geq p$*, and therefore the index set* $\bar{\Omega}_B^-$ *is just* $\{11, 13\}$. *The constraints* $\beta^i_{\ell,j} r_{\ell,j} \leq m - 1$ *for all* $i \in \bar{\Omega}_B^-$ *in* (6.2.9) *means that if* $r_{1,1} = 4$ *or* $r_{2,1} = 5$*, then* $r_{1,2}$ *cannot take the value 8 (a solution* $\bar{v}$ *of* (6.2.9) *must strictly dominate both scenarios* ω^{11} *and* ω^{13}*). In fact, the optimal value of* (6.2.9) *is reached when* $\bar{v}_1 = 4$*, the*

minimum element in Z_1, and $\bar{v}_2 = 9$, the smallest element in Z_2 allowed to combine with $\bar{v}_1 = 4$.

6.3.2 Continuous distributions

When the random variable has an infinite support, approximate p-efficient points can be obtained by combining sampling and restoration. We provide new theoretical insights on the link between the sample size N and feasibility for the continuous distribution, along the lines of [128, 181].

6.3.2.1 Sampling

Consider (6.3.2) for a given sample with N realizations $\omega^1, ..., \omega^N$, and let $\tilde{v}$ be a feasible point, computed for instance using h1. In general, $\tilde{v}$ is unfeasible for the continuous distribution, i.e., $\mathbb{P}[\omega \leq \tilde{v}] < p$, so $\tilde{v}$ is not an p-efficient point for the continuous distribution. To ensure that $\tilde{v} \in \mathcal{Z}$, the restoration step explained below can be used.

6.3.2.2 Restoration

Computing a Slater point for (6.3.2) is easy, because $\lim_{v\to\infty} \mathbb{P}[\omega \leq v] = 1$ and, hence, taking sufficiently large components for v^s ensures $\mathbb{P}[\omega \leq v^s] > p$ with the continuous distribution. Restoration is achieved by computing (the smallest) $\lambda \in (0, 1)$ such that $\mathbb{P}[\omega \leq v(\lambda)] \geq p$, where $v(\lambda) = \lambda\tilde{v} + (1 - \lambda)v^s$. In general this procedure requires a few interpolation steps only, depending on the required accuracy for feasiblity.

6.3.2.3 Linking the sample size to approximate p-efficiency

We first state a technical result relating feasibility for the continuous distribution with feasibility for problem (6.3.2). We just mention here that, differently from [128], our proof uses simultaneously Bernstein's and Hoeffding's bounds.

Lemma 6.3.3. *([4, Lemma 1].) Let $G : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^k$ be a mapping, $\omega \in \mathbb{R}^m$ a multivariate random variable and consider the probabilistic constraint $\psi(x) := \mathbb{P}[G(x, \omega) \leq 0] \geq p$, with feasible set $X(p) := \{x \in \mathbb{R}^n : \psi(x) \geq p\}$. For $N \geq 1$, let $\omega^1, ..., \omega^N$ be an i.i.d. sample of ω with approximate feasible set $M_q^N := \left\{x \in \mathbb{R}^n : \frac{1}{N}\sum_{i=1}^N \mathbb{1}_{G(x,\omega_i)\leq 0} \geq q\right\}$, where $1 > q > p$. For any $x \notin X(p)$, the following estimate holds true:*

$$\mathbb{P}[x \in M_q^N] \leq \exp\left(-N(q-p)^2 \max\left\{2, \frac{1}{2\psi(x)(1-\psi(x)) + \frac{2}{3}(q-p)}\right\}\right).$$

The lemma can be combined with a covering argument to determine the confidence level for a point that is feasible for problem (6.3.2) to be also feasible for the continuous probabilistic constraint. To do so, we define a grid which roughly covers the zone of interest. Here the advantage of being able to switch between Hoeffding's and Bernstein's bounds shows its full potential (the weaker form, using only Hoeffding's bound, is [128, Thm. 5]). Since many points in the grid will have a relatively low probability level, often the feasible region $X(p)$ has a very gully-shaped look, with "many" x having small $\psi(x)$ and the mapping $\psi(x)$ rapidly increases over a small range [134]. The advantage of using Bernstein's bound over Hoeffding's bound arises when $\psi(x) \leq \frac{1}{2} - \sqrt{\frac{1}{4} + (\frac{1}{3}(q-p) - \frac{1}{4})}$ or $\psi(x) \geq \frac{1}{2} - \sqrt{\frac{1}{4} + (\frac{1}{3}(q-p) - \frac{1}{4})}$. Therefore if many grid-points have a low probability level, a significant improvement could be obtained; the results in Figure 6.3 confirm this expectation.

Accordingly, in addition to the assumptions of Lemma 6.3.3, suppose the set $X \subseteq \mathbb{R}^n$ is compact. For $\eta > 0$, a collection of points $\mathcal{G} := \{x_i\}_{i=1}^K \subseteq X$ is called a $\mathfrak{g}$-*dominating η-lattice* if and only if for any $x \in X$ there exists $x_i \in \mathcal{G}$ with $\|x - x_i\| \leq \eta$ and $G(x, \omega) \leq 0$ implies $G(x_i, \omega) \leq 0$ almost surely.

The notion of a $\mathfrak{g}$-dominating η-lattice is very similar to the requirements of precedence in [175]. For the separable case ($G(x,\omega) = \mathfrak{g}(x) - \omega$), the construction in [128, Theorem 9] shows how such a $\mathfrak{g}$-dominating η-lattice can be set up. To this end, let l be the lower bound for $\mathcal{V}$ in Section 6.2.1. By compactness of X there exists $U \in \mathbb{R}^k$ such that $\mathfrak{g}(x) \leq U$ for all $x \in X$. We may therefore restrict our attention to the set $\{x \in X \ : l \leq \mathfrak{g}(x) \leq U\}$ without loss of generality. Now define the set of points $Y_j = \left\{ l_j + i\frac{(U_j - l_j)}{P}, i = 1, ..., P \right\}$ for each $j = 1, ..., k$ and $\mathcal{G} = \prod_{j=1}^{k} Y_j$. Then for any $y \in [l, U]$ we can find $y' \in \mathcal{G}$ such that $y \leq y'$ and $\|y - y'\| \leq \eta$. Indeed define the jth component of y' as $y'_j = \min_{w \in Y_j \, : w \geq y_j} w$. By construction $y' \geq y$ and $\left|y'_j - y_j\right| = y'_j - y_j \leq \frac{U_j - l_j}{P}$, which entails $\|y' - y\|_\infty \leq \max_{1 \leq j \leq k} \frac{U_j - l_j}{P}$. By equivalence of norms in $\mathbb{R}^k$, it is immediate that one can select P in such a way as to make the right-hand size smaller than any desired $\eta > 0$.

We now link the sample size N in problem (6.3.2) with feasibility of the resulting solutions for the probabilistic constraint with continuous distribution:

Theorem 6.3.4. *([4, Theorem 1].) With the assumptions and notation in Lemma 6.3.3, suppose that the set $X \subseteq \mathbb{R}^n$ is compact and ψ is Lipschitz continuous with constant L (w.r.t. the norm $\|\cdot\|$). Let $\mathcal{G} := \{x_i\}_{i=1}^{K}$ be a $\mathfrak{g}$-dominating η-lattice, for $\eta > 0$ such that $L\eta \in (0, q - p)$. For the approximate feasible set M_q^N given in Lemma 6.3.3, we have that*

$$\mathbb{P}[M_q^N \subseteq X(p)] \geq 1 - \sum_{j=1}^{K} \exp\left(-N(q - p - L\eta)^2 \max\left\{ 2, \frac{1}{2\psi(x_j)(1 - \psi(x_j)) + \frac{2}{3}(q - p - L\eta)} \right\} \right) \mathbb{1}_{x_j \notin M(p + L\eta)}.$$

Example 6.3.5. *A graphical illustration of Theorem 6.3.4 can be obtained by taking a set $\{x_j \in \mathcal{G} : \psi(x_j) \leq 0.1\}$ containing $K_1 \leq K$ points. Then $\mathbb{P}[M_q^N \subseteq X(p)] \geq 1 - K_1 \exp\left(-\frac{N(q-p-L\eta)^2}{\frac{18}{100} + \frac{2}{3}(q-p-L\eta)} \right) - (K - K_1)\exp\left(-2N(q - p - L\eta)^2\right)$. Note that when taking in the set $\psi(x_j) \leq 0.01$ the constant $\frac{18}{100}$ above is to be replaced by 0.0198. For the cash-matching problem [87], we have $K_1 \approx 0.98K$ and, for a given confidence level $1 - \delta$ and using concrete data $p = 0.8$, $q = 0.802$, $\eta = 0.001$, $L = 1$, we plotted the dependency of δ on N in Figure 6.3. The advantage of using Bernstein's over Hoeffding's bound can be noticed in the figure, which shows that the change roughly brings a gain of one order of magnitude on N. Further improvements can be achieved when for $0.98K$ points in the grid the probability is $\psi(x) < 0.01$, as shown by comparing the two Bernstein's plots in Figure 6.3.*

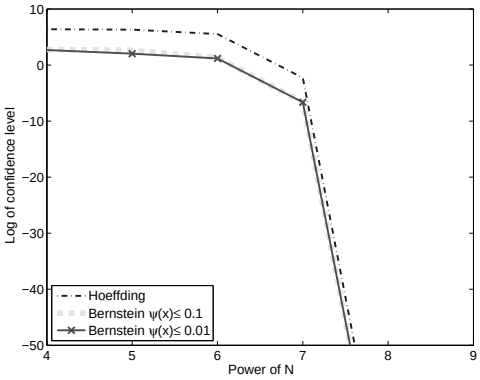


Figure 6.3: The size of N versus precision for the cash matching problem, with a logarithmic scale.

Recall that our approach is based on the iterative primal-dual generation of p-efficient points. Having addressed the primal concerns of how to efficiently approximate $d(u)$ in (6.2.4), we now turn our attention to the efficient generation of dual vectors u, by means of a bundle methodology.

6.4 Computing dual vectors: a bundle method detour

The dual problem (6.2.4) has a concave objective function φ which is nonsmooth at those points u^j having more than one x^j or v^j solving (6.2.7) or (6.2.8), respectively. Recall that in (6.2.5) the set $\mathcal{V}$ complicates the function calculation; otherwise, if φ was easy to compute, a *proximal algorithm* [137, 171] could be employed. At the kth-iteration and having a proximal stepsize $t_k > 0$, this method defines iterates as follows:

$$u^{k+1} := \arg\max_{u\in\mathbb{R}^m_+} \left\{ \varphi(u) - \frac{1}{2t_k} \left\| u - u^k \right\|^2 \right\} .$$

In our case, one iterate is as hard to compute as solving problem (6.2.4); the bundle approach presented below addresses this issue, paramount for computational efficiency.

6.4.1 On the importance of models

Bundle methods [29, Part II] replace the difficult function φ by a simpler *model* M^k, to be improved along iterations. With the exact function, the next iterate u^{k+1} always provides ascent, but now (depending on the model quality), there is no guarantee that $\varphi(u^{k+1})$ will be larger than $\varphi(u^k)$. Bundle methods separate iterates providing sufficient ascent into a special *center subsequence* $\{\hat{u}^k\}$. The limit points of this subsequence, also called sequence of *serious steps*, solve (6.2.4). A bundle algorithm of the proximal type defines iterates as below:

$$u^{k+1} = \arg\max_{u\in\mathbb{R}^m_+} \left\{ M^k(u) - \frac{1}{2t_k} \left\| u - \hat{u}^k \right\|^2 \right\} . \tag{6.4.1}$$

An example of the rule deciding when there is a serious step and the iterate becomes the new center $\hat{u}^{k+1}$ can be found in (6.4.13). We just recall here the optimality conditions characterizing u^{k+1}, the unique solution to the concave maximization problem (6.4.1)

$$u^{k+1} = \hat{u}^k + t_k \hat{s}^k \quad \text{with} \quad \hat{s}^k := p_1^k - p_2^k \ \text{ for } \ \left\{ \begin{array}{l} -p_1^k \in \partial(-M^k)(u^{k+1}) \\ p_2^k \in N_{\mathbb{R}^m_+}(u^{k+1}) . \end{array} \right. \tag{6.4.2}$$

The computational work involved in solving (6.4.1) depends on the specific model M^k. The general bundle theory [149] is very flexible in this sense, below we provide three possible choices.

Example 6.4.1 (Aggregate cutting-plane model: $M^k = \check{\varphi}^k$)**.** Perhaps the most natural choice is to replace the concave function φ by an outer approximation defined by linearizations, or cutting planes. This is the function

$$\check{\varphi}^k(u) := \min_{j\in J_k} L^j(u) \quad \text{where } L^j(u) := f(x^j) + \left\langle u, v^j - \mathfrak{g}(x^j) \right\rangle \qquad \begin{array}{l} \text{for } x^j \in X \text{ as in (6.2.7)} \\ \text{and } v^j \in \operatorname{conv}\mathcal{V} \text{ as in (6.2.8).} \end{array} \tag{6.4.3}$$

The index-set J_k gathers past linearization indices until iteration k; for instance all of them: $J_k = \{1, \ldots, k\}$ (in Section 6.5 more economical sets J_k, collecting less linearizations, are considered).

Taking the model $M^k = \check{\varphi}^k$ gives in (6.4.1) a convex quadratic programming problem (QP), easy to solve. Because the model is a piecewise affine convex function,

$$p_1^k = \sum_{j\in J_k} \alpha_j^{k+1} (v^j - \mathfrak{g}(x^j)) \text{ with } \alpha^{k+1} \in \Delta^{|J_k|} \text{ such that } M^k(u^{k+1}) = \sum_{j\in J_k} \alpha_j^{k+1} f(x^j) + \left\langle p_1^k, u^{k+1} \right\rangle . \tag{6.4.4}$$

In the bundle set J_k, *strongly active* indices correspond to active linearizations with positive weight:

$$j \in J_k \text{ such that } \alpha_j^{k+1} > 0 \text{ and } M^k(u^{k+1}) = L^j(u^{k+1}) . \tag{6.4.5}$$

For asymptotic analysis reasons, we assume that the number of strongly active indices is uniformly bounded in k. This natural property is ensured for instance by most active-set QP solvers, whose linearly

independent bases involve at most $m+1$ Carathéodory-like positive simplicial multipliers, regardless of the cardinality of J_k.

The aggregate cutting-plane model is used in the *Regularized Dual Method* [57, Sec. 4], taking a constant stepsize $t_k = t$ (coresponding to a fixed regularization parameter) and a full bundle of information $J_k = \{1, 2, \ldots, k\}$. □

The following model takes advantage of the sum-structure of the function φ.

Example 6.4.2 (Disaggregate cutting-plane model: $M^k = \check{h}^k + \check{d}^k$)**.** Each term has its own linearization

$$L_h^j(u) := f(x^j) + \big\langle -\mathfrak{g}(x^j), u \big\rangle \quad \text{and} \quad L_d^j(u) := \langle v^j, u \rangle \,.$$

The disaggregate model is the sum of the individual cutting-plane models, using separate index sets, J_k and $\tilde{J}_k$:

$$\check{h}^k(u) + \check{d}^k(u) \;\; := \;\; \min_{j \in J_k} L_h^j(u) \;\; + \;\; \min_{j \in \tilde{J}_k} L_d^j(u) \,.$$

Taking the disaggregate model gives again a convex QP subproblem (6.4.1), of larger size than the aggregate one, to account separately for the two bundles of information. In particular, the subgradient of this model at u^{k+1} is

$$p_1^k = \textstyle\sum_{j \in J_k} \alpha_j^{k+1} v^j - \sum_{j \in \tilde{J}_k} \tilde{\alpha}_j^{k+1} \mathfrak{g}(x^j) \text{ with } \begin{array}{l} \alpha^{k+1} \in \Delta^{|J_k|} \\ \tilde{\alpha}^{k+1} \in \Delta^{|\tilde{J}_k|} \end{array} \text{ such that } M^k(u^{k+1}) = \textstyle\sum_{j \in J_k} \alpha_j^{k+1} f(x^j) + \big\langle p_1^k, u^{k+1} \big\rangle \,, \tag{6.4.6}$$

and, hence, in this model there are strongly active indices (6.4.5) for each separate bundle. □

The third model takes the exact function h.

Example 6.4.3 (Partially exact model: $M^k = h + \check{d}^k$)**.** Only the difficult function d is modeled by cutting planes:

$$h(u) + \check{d}^k(u) := h(u) + \min_{j \in \tilde{J}_k} L_d^j(u) = \min_{x \in X} \Big\{ f(x) - \langle \mathfrak{g}(x), u \rangle \Big\} + \min_{j \in \tilde{J}_k} \langle v^j, u \rangle \,.$$

The subgradient for this model is

$$p_1^k = \sum_{j \in \tilde{J}_k} \tilde{\alpha}_j^{k+1} v^j - \mathfrak{g}(x^{k+1}) \text{ with } \begin{array}{l} h(u^{k+1}) = f(x^{k+1}) + \big\langle u^{k+1}, \mathfrak{g}(x^{k+1}) \big\rangle \\ \tilde{\alpha}^{k+1} \in \Delta^{|\tilde{J}_k|} \end{array} \tag{6.4.7}$$

and such that $M^k(u^{k+1}) = f(x^{k+1}) + \big\langle p_1^k, u^{k+1} \big\rangle$. Naturally, if neither f nor $\mathfrak{g}$ are linear or quadratic functions, the model $M^k = h + \check{d}^k$ no longer yields a QP and (6.4.1) becomes a general convex optimization problem. □

Regarding the quality of the various models above, from their definition it is straightforward that for all $u \in \mathbb{R}^m_+$

$$\check{\varphi}^k(u) \geq \check{h}^k(u) + \check{d}^k(u) \geq h(u) + \check{d}^k(u) \geq \varphi(u) \,, \tag{6.4.8}$$

i.e., the partially exact model is better than the disaggregate model, in turn better than the aggregate one. On the other hand, subproblem (6.4.1) becomes easier for the choice $M^k = \check{\varphi}^k$, and more difficult for $M^k = h + \check{d}^k$. It will be shown in Section 6.5 that (6.4.1) with $M^k = h + \check{d}_k$ corresponds to the Progressive Augmented Lagrangian method [56], which relaxes the constraint $\mathfrak{g}(x) \geq v$ in (6.2.6); see Example 6.5.3 for details.

6.4.2 The case of exact serious evaluations

As in [149], we suppose that the inaccurate evaluations in (6.3.1) have uniformly bounded errors:

$$\text{for all } u^j \in \mathbb{R}^m_+ \text{ the inaccuracy } \eta_{u^j} \text{ in (6.3.1) satisfies } \eta_{u^j} \le \eta \text{ for some } \eta \ge 0. \tag{6.4.9}$$

This does not require to know explicitly the oracle error bound η, its mere existence suffices (the bound is not used by the algorithm). By evaluating at $u = u^j$ the η_{u^j}-subgradient inequality in the last line of (6.3.1) we see that

$$\varphi_{u^j} \in [\varphi(u^j), \varphi(u^j) + \eta_{u^j}] \text{ approximates the exact value from above}. \tag{6.4.10}$$

We consider here only two situations. Either the oracle always delivers *exact* information ($\eta_{u^j} \equiv 0$), as in Examples 6.4.1 and 6.4.3; or the oracle delivers exact information only when there is a serious step, at points that become a center ($\eta_{u^j} > 0$, except for $\eta_{\hat{u}^k} \equiv 0$), as in Example 6.5.2 below, derived from Example 6.4.2. As a result, throughout this subsection the oracle error is null at serious steps: $\eta_{\hat{u}^k} \equiv 0$ (cf. (6.4.13)). The more general case will be the subject of Section 6.4.3, noting that for all the oracles and models (including those in Section 6.4.3) the model functions satisfy

$$\varphi(u) \le M^k(u) \quad \text{for all } u \in \mathbb{R}^m_+. \tag{6.4.11}$$

This property will be referred to as having an *upper model* (in the parlance of [147], minimizing the convex function $-\varphi$, the model is of lower type). Similarly, we shall say L^j is an *upper* linearization when

$$L^j(\cdot) \text{ is an affine function such that } \varphi(u) \le L^j(u) \quad \text{for all } u \in \mathbb{R}^m_+. \tag{6.4.12}$$

The solution of (6.4.1) with a model M^k satisfying (6.4.11) gives u^{k+1}. Given a parameter $\kappa \in (0,1)$, *a serious step* is declared and $\hat{u}^{k+1} := u^{k+1}$, when

$$\varphi_{u^{k+1}} \ge \varphi_{\hat{u}^k} + \kappa\, \mathtt{v}^k \;\text{ for } \mathtt{v}^k := M^k(u^{k+1}) - \varphi_{\hat{u}^k}, \text{ where } \varphi_{\hat{u}^k} = \varphi(\hat{u}^k) \text{ because } \eta_{\hat{u}^k} = 0. \tag{6.4.13}$$

Otherwise, the iteration is declared *null*, keeping $\hat{u}^{k+1} := \hat{u}^k$. The subgradient inequality for p_1^k in (6.4.2), combined with the definitions of $\hat{s}^k$ and p_2^k, ensures that $M^k(u^{k+1}) \ge M^k(\hat{u}^k) + t_k \left\|\hat{s}^k\right\|^2$. Then, (6.4.12) implies that (6.4.13) is an ascent test, checking increase in the function values:

$$\mathtt{v}^k \ge M^k(\hat{u}^k) + t_k \left\|\hat{s}^k\right\|^2 - \phi_{\hat{u}^k} \ge t_k|\hat{s}^k|^2 - \eta_{\hat{u}^k} = t_k \left\|\hat{s}^k\right\|^2 \ge 0 \quad \text{if } \eta_{\hat{u}^k} = \varphi_{\hat{u}^k} - \varphi(\hat{u}^k) = 0. \tag{6.4.14}$$

For future reference, note that the relations in (6.4.13) and (6.4.14) hold because the evaluation error is null at $\hat{u}^k$.

The *aggregate linearization*

$$A^k(u) := M^k(u^{k+1}) + \langle p_1^k, u - u^{k+1}\rangle \tag{6.4.15}$$

available after solving (6.4.1), is of the upper type (due to the definitions in (6.4.2)):

$$\varphi(u) \le A^k(u) \;\text{ for all }\; u \in \mathbb{R}^m_+. \tag{6.4.16}$$

The aggregate linearization is the highest outer approximation of φ that can be used without losing information, A^k condenses all the past information generated by the method and is the key behind the mechanism called *bundle compression* described in Example 6.5.1 below; see also [29, Ch. 10.3.2].

By combining the rightmost identities in (6.4.4)-(6.4.7) with (6.4.15) written for $u = 0$, we see that

$$A^k(0) = \begin{cases} \displaystyle\sum_{j \in J_k} \alpha_j^{k+1} c(x^j) & \text{with the aggregate and disaggregate models (Examples 6.4.1 and 6.4.2)} \\ c(x^{k+1}) & \text{with the partially exact model (Example 6.4.3).} \end{cases} \tag{6.4.17}$$

These relations are fundamental to show primal convergence. For dual convergence, two important objects, computable after solving (6.4.1), are the *aggregate gap* and the *Fenchel measure*, defined respectively by

$$\hat{e}_k := \mathrm{v}^k - t_k \left\|\hat{s}^k\right\|^2 \quad \text{and} \quad \phi^k := \hat{e}_k - \langle \hat{s}^k, \hat{u}^k \rangle \,. \tag{6.4.18}$$

As in (6.4.14), having upper models with exact serious evaluations implies that the gap is always nonnegative:

$$\hat{e}_k \geq -\eta_{\hat{u}^k} \,, \quad \text{if } \eta_{\hat{u}^k} = 0. \tag{6.4.19}$$

We now state some important primal-dual relations highlighting the role of these objects regarding convergence.

Lemma 6.4.4 (Primal and dual optimality certificates)**.** *([4, Lemma 2].) Suppose the oracle evaluations and the model satisfy, respectively,* (6.4.10) *and* (6.4.11)*. Associated with the model in* (6.4.1) *consider the primal pair*

$$(\hat{x}^{k+1}, \hat{v}^{k+1}) := \begin{cases} \sum_{j\in J_k} \alpha_j^{k+1} \left(x^j, v^j\right) & \text{generated with the aggregate model (Ex. 6.4.1)} \\ \left(\sum_{j\in J_k} \alpha_j^{k+1} x^j, \sum_{j\in \tilde{J}_k} \tilde{\alpha}_j^{k+1} v^j\right) & \text{generated with the disaggregate model (Ex. 6.4.2)} \\ \left(x^{k+1}, \sum_{j\in \tilde{J}_k} \tilde{\alpha}_j^{k+1} v^j\right) & \text{generated with the partially exact model (Ex. 6.4.3).} \end{cases} \tag{6.4.20}$$

The following holds:

(i) $\varphi(u) \leq \varphi(\hat{u}^k) + \eta_{\hat{u}^k} + \phi^k + \langle \hat{s}^k, u\rangle$ *for all* $u \in \mathbb{R}^m_+$.

(ii) The primal pair satisfies $(\hat{x}^{k+1}, \hat{v}^{k+1}) \in X \times \operatorname{conv} \mathcal{V} \subset X \times \mathcal{Z}$*, with*

$$\hat{v}^{k+1} \leq \mathfrak{g}(\hat{x}^{k+1}) + \hat{s}^k \quad \text{and} \quad f(\hat{x}^{k+1}) \leq \varphi(\hat{u}^k) + \eta_{\hat{u}^k} + \phi^k \,.$$

(iii) If $\hat{s}^k = 0$ *and* $\phi^k \leq 0$ *then* $\hat{u}^k$ *is an* $\eta_{\hat{u}^k}$*-solution to* (6.2.4) *and* $(\hat{x}^{k+1}, \hat{v}^{k+1})$ *is an* $\eta_{\hat{u}^k}$*-solution to* (6.2.2)*.*

Algorithm 3 Concave Proximal Bundle Method for Upper Models and Exact Serious Evaluations (PBM)

Step 0: Initialization. Select $\kappa \in (0,1)$ and $t_1 \geq t_{\mathtt{low}} > 0$. For $u^1 \in \mathbb{R}^m_+$ compute $h(u^1)$, $d(u^1)$ and subgradients (6.2.7) and (6.2.8). Choose a model $M^1 \geq \varphi$, stopping tolerances, Tol_ϕ, $\mathrm{Tol}_g \geq 0$ and set $\hat{u}^1 = u^1$, $k = 1$.

Step 1: Next iterate. Obtain u^{k+1} by solving (6.4.1).
Compute v^k as in (6.4.13), $\hat{s}^k = \frac{u^{k+1} - \hat{u}^k}{t_k}$ from (6.4.2), and ϕ^k from (6.4.18).

Step 2: Stopping test. If $\phi^k \leq \mathrm{Tol}_\phi$ and $\left\|\hat{s}^k\right\| \leq \mathrm{Tol}_g$, stop and return $\hat{u}^k$ and $(\hat{x}^{k+1}, \hat{v}^{k+1})$ from (6.4.20) as the solution.

Step 3: Oracle call. Compute an upper linearization L^{k+1} as in (6.4.12), for example using the exact values $h(u^{k+1})$, $d(u^{k+1})$ and respective subgradients (6.2.7) and (6.2.8).

Step 4: Ascent test. If (6.4.13) holds (serious step), make new calculations if needed, so that $\eta_{\hat{u}^{k+1}} = \eta_{u^{k+1}} = 0$. Set $\hat{u}^{k+1} = u^{k+1}$ and choose $t_k \geq t_{\mathtt{low}}$. If (6.4.13) does not hold (null step), set $\hat{u}^{k+1} = \hat{u}^k$ and choose $t_{k+1} \in [t_{\mathtt{low}}, t_k]$.

Step 5: Model. Choose a model satisfying $\varphi \leq M^{k+1} \leq \min\{L^{k+1}, A^k\}$.

Step 6: Loop. Set $k = k + 1$ and go back to Step 1.

Algorithm 3 describes the Proximal Bundle Method (PBM), when the model is of upper type and evaluation errors are null at serious steps: both (6.4.11) and (6.4.10) hold with $\eta_{\hat{u}^k} \equiv 0$.

By item (ii) of Lemma 6.4.4, the aggregate gradient and the Fenchel measure respectively estimate primal feasibility and the duality gap; the algorithm stops when those values are sufficiently small. Theorem 6.4.5 shows that (6.4.21), an asymptotic version of the conditions in item (iii), guarantees eventual solution of (6.2.4) and (6.2.2).

The need of new calculations mentioned in Step 4 in Algorithm 3 typically arises when the solution procedure in the oracle is organized so that first a coarse estimation is delivered, to check ascent. If the test declares a null step, the algorithm proceeds. Otherwise, the algorithm returns to the oracle requesting an exact calculation; examples of this situation are given by Steps 3' and 4' in Example 6.5.2 below, and also by the "coarse" and "fine" phases implemented for the on-demand oracles in Section 6.6.

The order of steps in Algorithm 3 is the usual one in bundle methods: first the new dual iterate is found in Step 1 and only in Step 3 the oracle finds corresponding primal points, solving the h and d problems in (6.2.4), (6.2.5). A variant with a primal-dual update is given in Example 6.5.3.

Regarding convergence, when PBM loops forever, there are two cases: either there is an infinite tail of null steps, or the algorithm generates infinitely many serious steps. These are the two mutually exclusive cases considered for the set K^∞ in Theorem 6.4.5, noting that always at least one of them has infinite cardinality when $k \to \infty$.

Theorem 6.4.5 (Primal and dual convergence for PBM)**.** *([4, Theorem 2].) Consider a primal problem* (6.2.2) *and its dual problem* (6.2.4) *such that the assumptions in Section 6.2.1 hold. Suppose that in Algorithm 3 the model satisfies* (6.4.11) *and the stopping tolerances are taken null (*$\mathrm{Tol}_\phi = \mathrm{Tol}_g = 0$*). If the oracle satisfies* (6.4.9) *and* (6.4.10) *and at serious steps there is no error evaluation (*$\eta_{\hat{u}^k} = 0$*) then*

$$\limsup_{k \in K^\infty} \phi^k \leq 0 \quad \textit{and} \quad \lim_{k \in K^\infty} \hat{s}^k = 0\,, \tag{6.4.21}$$

for an infinite iteration-set defined by

$$\begin{array}{lrcl} \textit{either} & K^\infty & := & \{k \geq \hat{k}\} \textit{ if after a last serious step at iteration } \hat{k} \text{ (6.4.13) } \textit{always fails} \\ \textit{or} & K^\infty & := & \{k : u^{k+1} \textit{ is declared serious in Step 4 }\}\,, \textit{ otherwise}\,. \end{array}$$

It follows that the primal subsequence $\{(\hat{x}^{k+1}, \hat{v}^{k+1})\}$ *always has limit points, and any of them solves* (6.2.2)*. When the center subsequence* $\{\hat{u}^k\}$ *has limit points, any of them solves* (6.2.4)*.* □

6.4.3 Handling inexact information at serious steps

So far, the models are built with exact oracle information at serious steps. Since exact evaluations involve computing a difficult p-efficient point, it is appealing to let evaluations at serious steps become exact only asymptotically. We now put in place an *on-demand accuracy* bundle method, able to handle oracles for which in (6.4.10) one may have $\eta_{u^j} > 0$ even when u^j becomes a serious step. The impact of this apparently innocuous modification is not so minor: if $\eta_{\hat{u}^k}$ is not null, the predicted increase in (6.4.14) (and the aggregate gap in (6.4.19)) may become negative. When this situation arises, the test (6.4.13) is meaningless because it no longer checks ascent. To handle this situation, Step 1 in Algorithm 3 must be suitably modified; this is the role of Step 1.2 of Algorithm 4 below, which declares noise as being "excessive" when the aggregate gap $\hat{e}_k$ becomes "too" negative.

The list below describes several possibilities for the oracle inaccuracy, the acronyms between parentheses refer to the corresponding methods benchmarked in Section 6.6:

- Having $\eta_{u^j} \equiv 0$ corresponds to the exact oracles in (6.2.7) and (6.2.8) (version 1 of BM and PAL).
- The case in which $\eta_{u^j} = 0$ if u^j yields a serious step gives the partially inexact Example 6.5.2 (versions 2 and 3 of BM and PAL).
- An asymptotically exact method drives the inaccuracy to zero for all points (version 4 of BM and PAL).

- A partially asymptotically exact algorithm drives $\eta_{\hat{u}^k} \to 0$ (version 5 of BM and PAL).

An inexact oracle designed to work in an *on-demand accuracy* mode returns function values whose error is smaller than η_{u^j}, sent as an input by the optimization procedure. For the functions h and d we now explain how to build such a mechanism while still ensuring (6.4.12), a property crucial to have upper models and show convergence. The starting idea is that, as both h and d in (6.2.4), (6.2.5) involve minimizing an objective function that is linear on the dual variable u, any feasible point realizes the η-subgradient inequality in (6.3.1). An on-demand oracle for d was devised in Heuristic h2, by stopping prematurely the solver for the d-problem in (6.2.5). For the function h, let $u^j \in \mathbb{R}^m$ and a bound $\eta_h^j \geq 0$ be given. When both f and $\mathfrak{g}$ are linear functions, a primal-dual Linear Programming solver is called. If the value η_h^j is set as stopping tolerance of the solver, the output will be a point $x^j \in X$ satisfying

$$\big[f(x^j) - \langle u^j, \mathfrak{g}(x^j)\rangle\big] - h(u^j) \leq \eta_h^j \,. \tag{6.4.22}$$

Taking $h_{u^j} := f(x^j) - \langle u^j, \mathfrak{g}(x^j)\rangle$ satisfies $h_{u^j} \in [h(u^j), h(u^j) + \eta_{u^j}]$, i.e., condition (6.4.10) for the function h. Similarly for (6.3.1), taking as subgradient $s_h^j := -\mathfrak{g}(x^j)$, and using (6.4.22),

$$\begin{aligned} h(u) &= \min_{x\in X}\Big\{f(x) - \langle u, \mathfrak{g}(x)\rangle\Big\} = \min_{x\in X}\big\{f(x) - \langle u^j, \mathfrak{g}(x)\rangle + \langle -\mathfrak{g}(x), u - u^j\rangle\big\} \qquad (6.4.23)\\ &\leq f(x^j) - \langle u^j, \mathfrak{g}(x^j)\rangle + \langle -\mathfrak{g}(x^j), u - u^j\rangle = h_u^j + \langle s_h^j, u - u^j\rangle =: L_h^j(u) \leq h(u^j) + \eta_h^j + \langle s_h^j, u - u^j\rangle \,. \end{aligned}$$

By (6.3.1), and in spite of inexactness, the linearization is still of the upper type. This follows from

$$\varphi(u) \leq \varphi(u^j) + \langle s^j, u - u^j\rangle \leq \varphi_{u^j} + \langle s^j, u - u^j\rangle =: L^j(u) \,,$$

ensuring (6.4.12) also with the inexact oracle. This validates the use of upper models M^k in Algorithm 4 below.

The important property (6.4.9) is straightforward for (6.4.23), because the inaccuracy is controlled by the bundle solver. We refer to [105, 147, 149, 208, 214] for different inaccurate oracles and ways to deal with inexactness.

We now describe the few modifications that need to be brought into PBM when the oracle delivers inexact information. The ascent test as well as the aggregate gap and Fenchel measure remain as in (6.4.13) and (6.4.18), respectively. The difference is that now the predicted increase and the gap can be negative. In particular, by (6.4.18),

$$\mathtt{v}^k \geq 0 \iff \hat{e}_k \geq -t_k \left\|\hat{s}^k\right\|^2 \,,$$

so to make (6.4.13) a meaningful ascent test ($\mathtt{v}^k \geq 0$) the *noise detection* step below checks if $\hat{e}_k < -\beta t_k \left\|\hat{s}^k\right\|^2$ for a parameter $\beta \in (0,1)$, as in [10, 149]. To *attenuate excessive noise*, the strategy proposed in [92, 105] can be adopted. Namely, increase the stepsize t_k and solve problem (6.4.1) changing neither the model nor the center. If no noise is detected, the predicted increase is nonnegative and the algorithm proceeds as Algorithm 3.

In Algorithm 4 the parameter `na` is used to block a decrease of the stepsize t_k when the iterate is declared null and noise attenuation steps had been done in Step 1.3 (otherwise, since Step 1.3 increases t_k, stepsize zigzagging could hinder the convergence process).

Regarding convergence, a difference with PBM is that, in addition to the usual serious and null steps dichotomy, now the algorithm can loop forever inside of Step 1, trying to attenuate noise. Assumption (6.4.9) ensures that the gap is bounded below, by (6.4.19). In this situation, having infinitely often $\hat{e}_k < -\beta t_k \left\|\hat{s}^k\right\|^2$ in Step 1.1 while driving $t_k \to \infty$ makes $\hat{s}^k$ eventually null and once more Lemma 6.4.4 applies. Naturally, the solution quality, both in terms of optimality and feasibility, depends on the asymptotic accuracy of the oracle at serious points. More precisely, on $\eta^\infty := \liminf \eta_{\hat{u}^k}$.

Algorithm 4 Concave Proximal Bundle Method for Upper Models with Inexact Oracles (PBM$_{\eta_{u^j}>0}$)

Step 0: Initialization. As in Step 0 of Algorithm 3, but with inexact values satisfying (6.3.1), and noise parameters $\mathtt{na} = 0$ and $\beta \in (0,1)$.

Step 1.1: Next iterate. As in Step 1 of Algorithm 3, computing also $\hat{e}_k$ from (6.4.18).

Step 1.2: Noise detection. If $\hat{e}_k \geq -\beta t_k \left\|\hat{s}^k\right\|^2$ go to Step 2 (noise is not too cumbersome).

Step 1.3: Noise attenuation. Set $t_{k+1} = 10t_k$, $\mathtt{na} = \mathtt{1}$, $M^{k+1} = M^k$, $\hat{u}^{k+1} = \hat{u}^k$, $k = k+1$, go back to Step 1.1.

Step 2: Stopping test. As in Step 2 of Algorithm 3

Step 3: Oracle call. Compute an upper linearization L^{k+1} as in (6.4.12), using oracle information satisfying (6.3.1), with u^{k+1} replacing u^j.

Step 4: Ascent test. If (6.4.13) is satisfied (serious step), set $\hat{u}^{k+1} = u^{k+1}$, $\mathtt{na} = 0$ and choose $t_k \geq t_{\mathtt{low}}$.
Otherwise (null step), set $\hat{u}^{k+1} = \hat{u}^k$ and choose $t_{k+1} \in [(1-\mathtt{na})t_{\mathtt{low}} + \mathtt{na}t_k, t_k]$.

Steps 5 and 6. As in Algorithm 3.

6.5 Relation with some dual methods from the literature

We consider the three model choices in Examples 6.4.1, 6.4.2 and 6.4.3, in the framework of Algorithms 3 or 4. We present new dual methods based on p-efficient points, and make some links with previous works in the literature.

Example 6.5.1 (Extending the Regularized Dual Decomposition of [57])**.** Consider PBM with aggregate cutting-plane model and exact evaluations. When M^k is the aggregate model in Example 6.4.1, and the oracle information is exact, Algorithm 3 is a standard proximal bundle method, with linearizations of the form

$$L^j(u) = \varphi(u^j) + \langle s^j, u - u^j\rangle = f(x^j) + \langle v^j - \mathfrak{g}(x^j), u\rangle \qquad \begin{array}{l}\text{for } x^j \in X \text{ as in (6.2.7)}\\ \text{and } v^j \in \operatorname{conv}\mathcal{V} \text{ as in (6.2.8).}\end{array} .$$

The Regularized Dual Decomposition [57] is a particular case of this variant, which fixes $t_k = t_0$ and sets in (6.4.3) the full index set, $J_k = \{1, \ldots, k\}$. The corresponding model update is $M^{k+1} = \min\{L^{k+1}, M^k\}$, which by (6.4.15) satisfies the conditions in Step 5. A difficulty with this update is that the size of the QPs (6.4.1) increases at each iteration. To keep the QP size controlled, the bundle can be reduced by introducing either a *selection* or a *compression* mechanism. The latter amounts to taking $M^{k+1} = \min\{L^{k+1}, A^k\}$ (similarly to the "generalized Frank-Wolfe rule" in [172, Eq. (3.31)]). This very economic model satisfies (6.4.11) and results in a QP with just two constraints, so each bundle iteration is fast, but many iterations may be needed to converge. By contrast, the selection mechanism keeps in the new model only *active* linearizations, as in (6.4.4):

$$M^{k+1} = \min\left\{L^{k+1}, \min\{L^j : j \in J_k \text{ such that } L^j(u^{k+1}) = M^k(u^{k+1})\}\right\} .$$

When compared with the compression technique, selection yields a less economic QP, but in general the additional time spent in solving (6.4.1) is compensated by a smaller number of iterations.

In view of Theorem 6.4.5, the Regularized Dual Decomposition [57] maintains its convergence properties for varying stepsizes satisfying $t_{\mathtt{low}} \leq t_k$ and with smaller QP subproblems, while improving its convergence speed. □

The aggregate model above uses exact information: in (6.4.10) the error η_{u^j} is always null. If the model M^k is chosen to be the disaggregate cutting-plane model, it is possible to avoid computing the

expensive d-information at null steps. We now explain how to implement this saving *without* impairing the convergence results in Theorem 6.4.5.

Example 6.5.2 (Two new on-demand accuracy methods)**.** Consider Algorithm 3 with a disaggregate partially inexact model. In Step 3, to provide an approximate value $\varphi_{u^{k+1}}$ satisfying (6.4.10), the oracle takes the exact value for $h(u^{k+1})$ and uses the cutting-plane value $\check{d}^k(u^{k+1})$ to replace the d-value. The new methods called BM2 and BM3 in the numerical Section 6.6 implement this variant.

Calculations are organized by modifying Steps 3 and 4 in Algorithm 3 as follows.

Step 3': First oracle call. Compute $h(u^{k+1})$ and its subgradient (6.2.7).
For the linearization L_d^{k+1} take the approximation $(d_{u^{k+1}}, s_d^{k+1}) := (\check{d}^k(u^{k+1}), \hat{v}^{k+1})$, available from (6.4.6).

Step 4': Ascent test and possible second oracle call. Declare a null step if $h(u^{k+1}) + d_{u^{k+1}} < \varphi(\hat{u}^k) + \kappa\, \mathtt{v}^k$. Otherwise, compute the exact d-information from (6.2.8) to obtain $\varphi(u^{k+1})$ and check if the (exact test) inequality $\varphi(u^{k+1}) \geq \varphi(\hat{u}^k) + \kappa\, \mathtt{v}^k$ is satisfied: if yes declare a serious step, otherwise a null step.

The inequality $\check{d}^k(u) \geq d(u)$ is always satisfied ($\check{d}^k$ is an upper model); so when the first test at Step 4' holds the (exact) ascent test fails and the iteration is declared null. The replacements Step 3' and Step 4' spare the the expensive d-computation at null points. At null steps, the model update in Step 5 takes $M^{k+1} = \check{h}^{k+1} + \check{d}^k$. At serious steps, $\eta_{u^{k+1}} = 0$ and the new exact linearization L_d^{k+1} enters the d-model: $M^{k+1} = \check{h}^{k+1} + \check{d}^{k+1}$. In both cases, the model satisfies (6.4.11) and, since (6.4.10) holds with $\eta_{\hat{u}^k} \equiv 0$, Theorem 6.4.5 applies.

For the disaggregate model it is also possible to put in place a selection/compression mechanism, proceeding separately for each term. The optimality conditions (6.4.2) for this specific model give an aggregate linearization that can be split into two functions, say A_h^k and A_d^k. Then Step 5 can take any cutting-plane model satisfying

$$M^{k+1} = \check{h}^{k+1} + \check{d}^{k+1} \text{ with } h \leq \check{h}^{k+1} \leq \min\Big\{L_h^{k+1},\, A_h^k\Big\} \text{ and } d \leq \check{d}^{k+1} \leq \min\Big\{L_d^{k+1},\, A_d^k\Big\}.$$

The new methods BM4 and BM5 in the numerical Section 6.6 use the disaggregate model with inexact oracle information in a manner similar to BM2 and BM3, but in the framework of Algorithm 4.□

The next example is the *Progressive Augmented Lagrangian* introduced in [56]. This variant of Algorithm 3 switches the original dual-primal order for the updates (Steps 1 and 3), defining first primal iterates.

Example 6.5.3 (Extending the Progressive Augmented Lagrangian algorithm of [56])**.** We now focus on PBM with a partially exact model, M^k is

$$M^k(u) = h(u) + \check{d}^k(u) = \min_{x\in X}\Big\{f(x) - \langle \mathfrak{g}(x), u\rangle\Big\} + \min_{j\in\tilde{J}_k}\langle v^j, u\rangle\,,$$

given in Example 6.4.3. Following the development in [56] we now make the relation between PBM and the augmented Lagrangian method for (6.2.2); see also [119]. We start by rewriting the cutting-plane model $\check{d}^k$ as an optimization problem:

$$\check{d}^k(u) = \min_{j\in\tilde{J}_k}\langle v^j, u\rangle = \left\{\begin{array}{ll}\min_v & \langle v, u\rangle \\ & v \in V_k \text{ from (6.2.6)}\end{array}\right. = \left\{\begin{array}{ll}\min & \langle \sum_{j\in\tilde{J}_k}\alpha_j v^j, u\rangle \\ \text{s.t.} & \alpha \in \Delta^{|\tilde{J}_k|}\,.\end{array}\right.$$

This notation is useful to rewrite the partially exact model M^k in the form

$$M^k(u) = \min_{x\in X,\alpha\in\Delta^{|\tilde{J}_k|}}\left\{f(x) - \langle \mathfrak{g}(x), u\rangle + \langle \sum_{j\in\tilde{J}_k}\alpha_j v^j, u\rangle\right\},$$

showing that the model is in fact the dual function associated with problem (6.2.6). More precisely, letting u denote the multiplier associated with the constraint $\mathfrak{g}(x) \geq v$, we see that

$$M^k(u) = \min_{x \in X, \alpha \in \Delta^{|\tilde{J}_k|}} L(x, \alpha; u) \quad \text{for } L(x, \alpha; u) := f(x) + \langle u, \sum_{j \in \tilde{J}_k} \alpha_j v^j - \mathfrak{g}(x) \rangle \, .$$

Therefore, the (negative of the concave) model has subgradients of the form $\mathfrak{g}(x) - \sum_j \alpha_j v^j$ for any pair (x, α) solving the minimization above. In particular,

$$M^k(u^{k+1}) = L(x^{k+1}, \alpha^{k+1}; u^{k+1}) = \min_{x \in X, \alpha \in \Delta^{|\tilde{J}_k|}} L(x, \alpha; u^{k+1}) \text{ for } (x^{k+1}, \alpha^{k+1}) \text{ from (6.4.7).}$$

Regarding Algorithm 3, this model gives for (6.4.1) in Step 1 the concave subproblem

$$u^{k+1} \text{ solves } \max_{u \geq 0} \Big\{ M^k(u) - \frac{t_k}{2} \left\| u - \hat{u}^k \right\|^2 \Big\} \equiv \max_{u \geq 0} \min_{x \in X, \alpha \in \Delta^{|\tilde{J}_k|}} \Big\{ L(x, \alpha; u) - \frac{t_k}{2} \left\| u - \hat{u}^k \right\|^2 \Big\} .$$

The argument in the right hand side formulation is the *regularized Lagrangian* from [56, Eq. (25)]. By strict concavity with respect to u and compactness of X, the triplet $(x^{k+1}, \alpha^{k+1}, u^{k+1})$ is a saddle point for the regularized Lagrangian and, in particular,

$$\max_{u \geq 0} \min_{x \in X, \alpha \in \Delta^{|\tilde{J}_k|}} \Big\{ L(x, \alpha; u) - \frac{t_k}{2} \left\| u - \hat{u}^k \right\|^2 \Big\} = \min_{x \in X, \alpha \in \Delta^{|\tilde{J}_k|}} L(x, \alpha; u^{k+1}) - \frac{t_k}{2} \left\| u^{k+1} - \hat{u}^k \right\|^2 .$$

The expression of p_1^k from (6.4.7) together with the normal element definition for p_2^k gives in (6.4.2) that

$$u^{k+1} := \max\Big(0, \, \hat{u}^k + t_k \Big(\sum_{j \in \tilde{J}_k} \alpha_j^{k+1} v^j - \mathfrak{g}(x^{k+1}) \Big) \Big) \quad \text{for } (x^{k+1}, \alpha^{k+1}) \text{ from (6.4.7),} \tag{6.5.1}$$

which highlights the primal-dual feature of this variant: to make the dual update, the primal points x^{k+1} and $v^{k+1} = \sum_{j \in \tilde{J}_k} \alpha_j^{k+1} v^j$ need to be available.

The Augmented Lagrangian perspective from [56] reveals that the primal points can be computed by solving, either the dual problem (6.4.1) written with the partially exact model, or a problem on primal variables, involving the Augmented Lagrangian associated with (6.2.6). More precisely, consider

$$L_k(x, \alpha; u) := f(x) + \langle u, G(x, \alpha; u) \rangle + \frac{t_k}{2} \left\| G(x, \alpha; u) \right\|^2$$

where we defined

$$G_i(x, \alpha; u) := \max \left(-\frac{u_i}{t_k}, \sum_{j \in \tilde{J}_k} \alpha_j v_i^j - \mathfrak{g}_i(x) \right) \quad \text{for } i = 1, \ldots, m.$$

Taking the derivatives and using the definition above, it is easy to see that

$$\frac{\partial L_k(x, \alpha; u)}{\partial (x, \alpha)} = \frac{\partial L(x, \alpha; u + t_k G(x, \alpha; u))}{\partial (x, \alpha)} \, .$$

Since in addition

$$u + t_k G(x, \alpha; u) = \max\Big(0, \, u + t_k \Big(\sum_{j \in \tilde{J}_k} \alpha_j v^j - \mathfrak{g}(x) \Big) \Big) \, ,$$

together with (6.5.1) we see that

$$\frac{\partial L_k(x^{k+1}, \alpha^{k+1}; \hat{u}^k)}{\partial (x, \alpha)} = \frac{\partial L(x^{k+1}, \alpha^{k+1}; u^{k+1})}{\partial (x, \alpha)} \, .$$

This means that

$$\text{the pair } (x^{k+1}, \alpha^{k+1}) \text{ solves } \min_{x\in X, \alpha\in\Delta^{|\tilde{J}_k|}} L_k(x,\alpha;\hat{u}^k) = \min_{x\in X, \alpha\in\Delta^{|\tilde{J}_k|}} L(x,\alpha;u^{k+1})\,,$$

and, hence, solving the left hand side problem above gives the desired primal points, to be used in (6.5.1) to make the dual update. Accordingly, Algorithm 3 with the partially exact model can be enhanced as follows:

Step 1': Next primal and dual iterates. Obtain (x^{k+1}, α^{k+1}) by solving $\min_{x\in X, \alpha\in\Delta^{|\tilde{J}_k|}} L_k(x,\alpha;\hat{u}^k)$, and compute u^{k+1} as in (6.5.1).
Compute v^k as in (6.4.13), $\hat{s}^k = \frac{u^{k+1}-\hat{u}^k}{t_k}$ from (6.4.2), and ϕ^k as in (6.4.18).

Step 3': Oracle call of d. Compute the linearization L_d^{k+1} using the exact information from (6.2.8).

In Step 3', the oracle only delivers information on d because the model $M^k = h + \check{d}^k$ has no need of linearizations for h. In view of (6.4.7), once the primal point x^{k+1} is available in Step 1', both the exact function value and a subgradient for h are straightforward to compute. Since only exact information is used in this variant (either via the oracle or directly from h), convergence follows from Theorem 6.4.5.

Algorithm 3 with the modified Steps 1' and 3' corresponds to the Progressive Augmented Lagrangian algorithm in [56], with the additional flexibility of allowing for varying stepsizes and bundle selection or compression. Specifically, to manage the bundle size, as only a bundle for d is defined, instead of (6.4.15), the aggregate linearization is the affine function $\check{d}^k(u^{k+1}) + \langle v^{k+1}, u - u^{k+1}\rangle = \langle v^{k+1}, u\rangle$.

Finding a uniform bound η as in (6.4.9) with the partially inexact model may be delicate if the dual sequence becomes unbounded (in this case, Step 4' needs exact oracle evaluations until a serious iteration can be declared).□

6.6 Numerical Comparison of Solvers

We compare 15 methods implemented in C++, obtained by combining the three dual models with five oracles.

Instances We consider seven problems listed in Table 6.2, they are linear programs with bilateral probabilistic constraints fitting (6.2.1), with $\omega \in \mathbb{R}^m$ a centered multivariate Gaussian random variable, that is

$$X = \{x \in [\underline{x}, \overline{x}] \subset \mathbb{R}^n : Ax \le b\}\,, \quad \text{and} \quad \mathbb{P}[a^r + A^r x \le \omega \le B^r x + b^r] \ge p\,.$$

Table 6.2: Size of problems in the benchmark. Here # A stands for the number of rows in matrix A .

Problem	n	# A	m	p	description
CM	3	1	15	0.9	cash matching [87]
Ain48	672	1296	48	0.8	cascaded reservoir management [7]
Isr48	566	268	48	0.8	cascaded reservoir management[7]
Isr96	566	172	96	0.8	cascaded reservoir management[7]
Isr168	566	28	168	0.8	cascaded reservoir management[7]
PTP1	2000	40	50	0.9	probabilistic transportation problem [128]
PTP2	2000	40	50	0.9	probabilistic transportation problem [128]

For each problem, we considered 7 instances, corresponding to $N \in \{50, 100, 250, 500, 1000, 2000, 5000\}$.

Oracles The calculations for h in (6.2.4) involve solving a simple LP problem, so the h-oracle is exact. For the d-oracle (6.2.5), the oracle error bound in (6.4.9) is $\eta^{\infty} = 10^{-4}$. We defined several on-demand accuracy versions: at iteration k the oracle receives u^{k+1}, a target $\texttt{tar}^k$ (typically some value greater than $\varphi_{\hat{u}^k}$, for instance the right hand side term in the ascent test (6.4.13)), and a bound $\texttt{gap}^k$ for the inaccuracy $\eta_{u^{k+1}}$.

To compute the oracle output, calculations are split into a *coarse* and a *fine* phase:

1. The coarse phase uses Heuristic h1 or h2 from Section 6.3 to define an estimate d_u^{k+1} for $d(u^{k+1})$. This phase is meant to be fast (only a few minutes). If the target is reached, i.e., $h(u^{k+1}) + d_u^{k+1} < \texttt{tar}^k$, the oracle returns the information to the bundle method and the algorithm proceeds in Step 4 to test for ascent. Otherwise, if $h(u^{k+1}) + d_u^{k+1} \geq \texttt{tar}^k$, the oracle passes to the next phase.
2. The fine phase computes better estimates by solving problem (6.3.2) until reaching either a relative gap inferior to $\texttt{gap}^k$ or the one hour CPU time limit.

Inexact Oracles

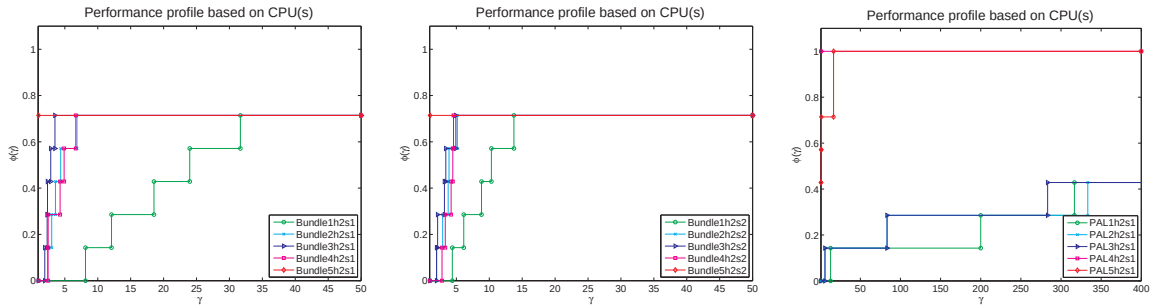
- 1: $\texttt{tar}^k = -\infty$ and $\texttt{gap}^k = 10^{-4}$. This oracle always passes to the fine phase and corresponds roughly to an exact PBM or to the *Progressive Augmented Lagrangian - PAL*, except when the 1h time limit is reached and the oracle outputs inexact information.
- 2: $\texttt{tar}^k = \varphi(\hat{u}^k)$ and $\texttt{gap}^k = 10^{-4}$. The d-oracle information is exact (or at least more accurate) at iterates which provide some ascent with respect to the threshold $\varphi(\hat{u}^k)$: $\eta_{\hat{u}^k} = 0$.
- 3: $\texttt{tar}^k = \varphi(\hat{u}^k) + \kappa\, \texttt{v}^k$ and $\texttt{gap}^k = 10^{-4}$. Similar to variant 2.
- 4: $\texttt{tar}^k = -\infty$ and $\texttt{gap}^k = \min\left\{0.5, \frac{0.01\texttt{v}^k}{k}\right\}$. Since the oracle error vanishes because $\texttt{gap}^k$ goes to zero, the d-oracle information is asymptotically exact (at all iterations).
- 5: $\texttt{tar}^k = \varphi(\hat{u}^k) + \kappa\, \texttt{v}^k$ and $\texttt{gap}^k = \min\left\{0.5, \frac{0.01\texttt{v}^k}{k}\right\}$. The d-oracle information is asymptotically exact at serious steps.

For all the methods the total CPU time limit is 48 hours, with $\kappa = 0.1$ to test for descent in (6.4.13), $\beta = -1.0$ to test for noise in Step 1.2 of $\text{PBM}_{\eta_{u^j}>0}$, and no bundle compression/selection. The rule to update the stepsize t_k is the *poorman* formula in [29, Sec. 10.3.3]. To solve (6.4.1) we use the dual simplex QP method of CPLEX 12.4. Computations were carried out on Intel Xeon X5670 westmere computer cluster with 8 Gb of reserved memory.

To refer to the many possible combinations, we adopt as mnenomics to append to BM1 (or BM2, BM3, etc) first the heuristic name and then a letter, a or d, to identify the aggregate or disaggregate cutting-plane model. For instance, when BM2 uses Heuristic h1 and an aggregate model, it is referred to as BM2h1a. Its counterpart using Heuristic h2 and disaggregate model is BM2h2d. Quite similarly PAL5 refers to the partially exact dual model and the use of oracle Strategy 5 (asymptotically exact at serious steps) described above. We exploited the scenario preprocessing method from [116] as it consistently reduced the total number of scenarios by up to 70%.

Results The quality of $\hat{x}^{k_{stop}+1}$ from (6.4.20) (iteration k_{stop} triggered the stopping test) can be checked assessing feasibility w.r.t. the continuous distribution problem, by computing $\mathbb{P}[a^r + A^r x^{k_{stop}+1} \leq \omega \leq B^r x^{k_{stop}+1} + b^r] \geq p$ using the code [75]. Instances Isr168, PTP1 and PTP2 require more than 2000 scenarios for the solution to be feasible for the continuous problem. Problems involving a large number of scenarios can only be solved when exploiting the inexact oracles at its full potential, i.e., using Strategy 5. This is seen by making performance profiles on CPU times, [61], with the proportion of problems that solved by each method, within a factor of the time required by the best algorithm (ordinate $\phi(\gamma)$

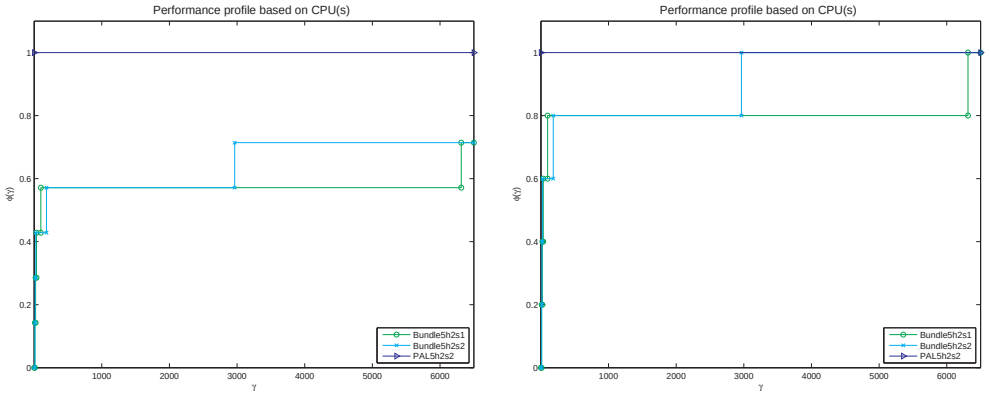
and abscissa γ in the profiles). When it comes to speed, the leftmost ordinate value gives the probability of each method to be the fastest in the benchmark, while robustness is seen by the rightmost ordinate value, with the proportion of problems solved by each method.



(a) Aggregate Model (b) Disaggregate Model (c) Partially Exact Model

Figure 6.4: Performance profiles for various model variants and oracle strategies.

Figure 6.4 shows how the different model variants perform depending on the oracle strategy. A first immediate observation is that the 5th oracle strategy is the most succesful, independently of the model variant. Most notably, while the PAL methods fail on nearly half the instances with oracles strategies 1 to 3, all instances were solved with oracles 4 and 5, i.e., as soon as inexact computations for p-efficient points were integrated in the oracle strategy. This phenomenon is not present in the other bundle methods, where all oracle strategies roughly solved the same percentage of instances within the time limit. Both the aggregate and disaggregate bundle methods managed to solve all instances with 1000 scenarios or less, regardless of the oracle strategy. In contrast, the partially exact dual model did not manage to solve all the 1000 scenario instances for oracle strategies 1 to 3.



(a) All (b) Easy

Figure 6.5: Oracle strategies 5 and 3 on all the instances (left) and on an "easy" subset, $N \leq 1000$ (right).

The next observation refers to Figure 6.5, where the importance of the relation (6.4.8) is striking: the partially exact model systematically brings very significant improvements This model has two potential drawbacks, though: it needs an exact oracle for h, and it can make too difficult the bundle subproblem (6.4.1), if h is a general concave function. This observation is backed up by Table 6.3, showing that 97 % of the CPU time is spent in the d-oracle (mean over all methods and instances). This cost levels off as the dual model model improves and the oracle variant exploits more "inexactness".

Dual model/Oracle variant	1	2	3	4	5
Aggregate	99.9	99.7	99.7	95.2	92.9
Disaggregate	99.5	99.4	99.3	96.0	90.6
Partially Exact	98.2	97.6	97.7	91.6	86.6

Table 6.3: Percentage of the total CPU time spend in the d-oracle

Concluding Remarks

A unified framework for dual approaches based on p-efficient points was presented. Thanks to the adopted bundle perspective, it is possible to compute approximate p-efficient points without losing precision in the solution. The numerical experiments highlight the interest of such inexact oracles, especially when N is large. The need of having a sufficiently large number of realizations to ensure feasibility was also demonstrated. As line of future work, possibly escaping the convenient setting (6.4.11), we mention the development of solvers dealing with oracles that progressively increase N along iterations, all the while using bundle management in order to eliminate previously generated points that are not p-efficient.

Chapter 7

Concluding remarks and pespectives

This document presents an excerpt from the research results that I have obtained since I received my PhD in January 2011. Chapter 1 highlights my research activities. To that end, the chapter was split into subfields of research pointing out my contributions on convex nonsmooth optimization, convex nonsmooth MINLP, nonsmooth DC programming, stochastic programming with recourse, and chance-constrained programming.

Chapters 2, 3, 4, 5 and 6 presented five of my recent papers. Each one of which illustrates a different aspect of my research. Two of my works on (deterministic) nonsmooth optimization were presented in Chapters 2 and 3. Chapter 2 was dedicated to the target radius method, a level-bundle like algorithm for solving convex nonsmooth programs [50]. Extensions of the method to deal with particular class of nonlinearly constrained and bilevel otimization problems were investigated. Nonsmooth nonconvex programs were considered in Chapter 3, where a new family of DC algorithms with inertial force was proposed to compute critical points of DC programs. The content of the chapter was extracted from the manuscript [51] that deals with nonconvex denoising models.

By dealing with a real-life application from the field of energy, Chapter 4 combined two-stage stochastic linear programming, scenario reduction techniques, and a bundle method algorithm to deal with the Brazilian natural gas network planning problem. The content of that chapter is the article [32].

Chapter 5 presented my work on eventual convexity of Copulæ structured chance-constrained programming. The material was extracted from the article [11]. The discussion on optimization problems with probability constraint was extended to Chapter 6 where the work [4] on chance-constrained programming via (inexact) p-efficient points was considered

7.1 General perspectives

Nonsmooth optimization, stochastic programming, and energy problems are always keywords in my research perspectives. To these, I would like to add more two items, one of practical and the other of a theoretical nature: climate and variational analysis.

Climate. Methodologies for creating multi-scale optimization models (spatial and temporal) that tackle energy system evolution issues in the short and long terms are of key importance to the industry of energy. The importance of taking into account climate models and new legislation on the energy system evolution becomes evident nowadays. A more accurate modeling gives rise to optimization problems that evolve and become more complex.

Regarding this area of application, my assignment in 2017 to CMA Mines ParisTech was motivated by and leads me to focus my work around optimization issues in the field of energy and climate, which has gained significant attention in both scientific and industrial spheres. In addition, these issues have some

technical characteristics that interest me and, moreover, take part in the modeling activities associated with the *ParisTech Chair on Modeling for Sustainable Development*, in which CMA is recognized for its expertise in long-term approaches.

I have already started working on this subject with my colleagues from CMA.

Variational analysis. As mentioned above, accurate optimization models taking into account climate and new legislation to tackle energy system evolution become more and more complex. Very often, not only differentiability assumptions are absent, but also convexity. These issues give rise to optimization problems possessing a high level of difficulty, which is sometimes prohibitive in the large-scale setting. Therefore, new techniques from variational analysis and computational mathematics are required to improve solution methods in this setting.

In my future research, the intention is to work on extensions of regularity concepts appropriate for studying stability (the "radius of good behaviour") of solutions to optimization problems, particularly those related to the stability of nonconvex stochastic problems and also optimization problems with equilibrium constraints, when standard assumptions are not satisfied (e.g. standard constraint qualifications). The outcomes will have an impact on enhancing convergence of numerical methods and facilitating the post-optimal analysis of solutions. An initial work on this direction is the manuscript [201] that investigates constraint qualification facilitating the verification and computation of Bouligand-stationary points of certain nonconvex nonsmooth programs.

7.2 Specific perspectives

In terms of research perspectives, there are several direct and indirect extensions of my works on stochastic programming with recourse, chance-constrained programming, and nonsmooth optimization (both convex and nonconvex cases). Some future works in each of one these fields are listed below.

7.2.1 Stochastic programming with recourse

Multistage stochastic programs explicitly model a series of decisions interplayed with partial observation of the uncertainties, which are approximated by scenario trees. This yields large-scale mathematical programming problems that can only be handled by specialized algorithms that employ decomposition techniques and very often sampling, as the stochastic dual dynamic programming (SDDP) [157]. In general terms, the SDDP is a cutting-plane based-method, which is well-known for presenting slow convergence when dealing with large-scale problems. Several strategies have been proposed to overcome this issue: aggregation of state-variables, cut-selection strategies, and scenario reduction. Recent attempts to accelerate the SDDP convergence by employing regularization techniques have been investigated in [18] and [199]. The main idea is to control trial points in the forward step as an attempt to construct cuts in favorable regions of the problem (improving thus the method's efficiency). These regularized works require the solution of more complicated subproblems in the forward step, which may entail additional difficulties when the number of decision variables per stage is large. To cope with this more general setting, other (computational cheaper) strategies to tackle multistage programs should be investigated. The intention to study more straightforward regularization strategies for the SDDP algorithm. A first attempt is to define trial points in the forward step as Chebyshev centers of certain polyhedrons issued by the cutting-plane approximation of the recourse functions. Such a strategy yields regularized linear subproblems that are not more difficult than the unregularized ones of SDDP. This is a subject under investigation with my PhD student Felipe Beltrán and colleagues from (UFSC, Brazil).

7.2.2 Stochastic programming with chance constraints

Concerning optimization problems with chance constraints, the topics of research envisaged in the short term are (i) DC approximations of probability functions and (ii) study and applications of Copulæ to model climate variables in energy management problems. For instance, instead of using forecasts for wind or photovoltaic power generation in a smart-grid, we could model the uncertainties and their inter-independence by employing specialized Copulæ. The estimation of an appropriate copula and its variational analysis (eventual convexity, differentiability) are not trivial tasks in general. This explains in general terms the item (ii). Concerning the item (i), the plan is to investigate probability functions of the form

$$\mathbb{P}[c_1(x,\omega) - c_2(x,\omega) \leq 0], \tag{7.2.1}$$

with c_1, $c_2 : \mathbb{R}^n \times \Omega \to \mathbb{R}$ convex (w.r.t argument x) but possibly nonsmooth functions. This kind of probability function appears, for instance, in chance-constrained optimal power flow problems modeling power-demand uncertainties, and in gas transport management [79].

Given $t > 0$ a small parameter, the (discontinuous) characteristic function $\mathbf{1}_{[0,+\infty)}(z)$ can be approximated by the DC function $[z+t]^+/t - [z]^+/t$. This allows the following approximation

$$\begin{aligned} \mathbb{P}[c_1(x,\omega) - c_2(x,\omega) \leq 0] &= \mathbb{E}[\mathbf{1}_{[0,+\infty)}(c_2(x,\omega) - c_1(x,\omega))] \\ &\approx \frac{1}{t}\mathbb{E}[\max\{c_2(x,\omega)+t,\, c_1(x,\omega)\}] - \frac{1}{t}\mathbb{E}[\max\{c_2(x,\omega),\, c_1(x,\omega)\}] \end{aligned}$$

which is a DC function. The expectation $\mathbb{E}[\cdot]$ can be approximated via Monte-Carlo simulation by considering a fixed sample of scenarios randomly generated accordingly to the distribution of ω. Hence, the problem of minimizing a convex (or DC) function subject to a convex set and chance constraint (7.2.1) can be efficiently approximated by a (DC-constrained) DC program. This motivates, from the variational-analysis point of view, the study of nonsmooth DC constraints to investigate proper constraint qualifications facilitating the verification and computation of stationary points, as well as the study of calmness yielding exact (DC) penalization functions to general DC problems.

7.2.3 Convex nonsmooth optimization

Large problems in stochastic programming with recourse are heavily structured, often amenable to parallel computing by standard decomposition schemes (e.g. by scenarios, by production units, or even both). For instance, two-stage programs with finitely many scenarios can be written as

$$\min_{x \in X} f(x), \quad \text{with} \quad f(x) := \sum_{i=1}^{m} f^i(x)\,,$$

where each component function $f^i : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ is convex. Existing optimization algorithms exploiting this decomposability are all synchronous: given a point x_k, m machines (oracles) deliver $(f^i(x_k),\, g_k^i \in \partial f^i(x_k))$ yielding $f(x_k) = \sum_{i=1}^m f^i(x_k)$ and a subgradient $\sum_{i=1}^m g_k^i = g_k \in \partial f(x_k)$ so that a master program computes a new trial point x_{k+1}. It is then evident that the master program may be idle for a long time, waiting all the m machines to return with (synchronous) information. In unit-commitment problems from energy management, some component functions f^i are assessed by solving relatively large mixed-integer linear programming problems, whereas other components can be easily computed by solving (much simpler) linear programming problems. Therefore, until all machines respond, not only the master program is idle but also the machines assigned to the easy-to-evaluate component functions f^i. If, instead, the master program processes a new trial point as soon as a single machine f^i responds, neither oracles/machines nor the master program becomes idle and the optimization process is boosted.

The goals in a short-term research plan are (i) to explore this idea and develop a fully asynchronous bundle method for difficult nonsmooth convex optimization problems with additive structure; and (ii) to investigate randomized variants of the method yielding *almost surely convergence*. To date, neither fully asynchronous nor randomized bundle method exists in the literature.

7.2.4 Nonconvex nonsmooth optimization

As argued above, the use of more sophisticated models to represent real-life problems comes at a price: complex (large-scale, nonsmooth, nonconvex) optimization models. To tackle such problems, the plan is to continue the work started in 2017 on DC programming and investigate more general nonsmooth nonconvex optimization problems.

Concerning general DC programs of the form

$$\min_{x \in X} \; f_1(x) - f_2(x) \quad \text{s.t.} \quad c_1(x) - c_2(x) \leq 0, \tag{7.2.2}$$

where X and all involved functions are convex, the plan is to investigate (in addition to what was commented in the Subsection 7.2.2) specialized primal-dual methods for efficiently handling this kind of problems. My first intention is to employ generalized augmented Lagrangian functions yielding zero-duality gap between the dual and primal optimal values. For the dual problem that consists in maximizing the (hard-to-evaluate) concave dual function, the use of the arsenal of inexact bundle methods developed in [149] might be employed to design a specialized bundle algorithm for handling generalized Lagrangians. Concerning the dual function, whose value and subgradient are computed by globally solving a convex-contrained DC program, the intention is to combine the ideas of the recent manuscript [51] with randomized strategies in order to compute critical points of good quality.

Another subject plan worth investigating is the problem of minimizing a lower C^2 function $f : \mathbb{R}^n \to \mathbb{R}$ over a convex set X. If f is lower C^2, then there exists a constant ρ such that $f(x) + \rho \|x\|^2 /2$ is convex on X. Hence,

$$f(x) = f(x) + \frac{\rho}{2} \|x\|^2 - \frac{\rho}{2} \|x\|^2$$

is indeed a DC function and DC algorithms could be employed to compute a stationary point of

$$\min_{x \in X} \; f(x)$$

as long as the constant ρ is known. Since ρ is in general unknown, the plan is to study strategies to estimate ρ that, when combined with DC algorithms, may yield practical algorithms to this class of problems.

Bibliography

[1] W. van Ackooij. "Chance Constrained Programming: with applications in Energy Management". PhD thesis. École Centrale Paris, 2013, p. 250.

[2] W. van Ackooij. "Decomposition approaches for block-structured chance-constrained programs with application to hydro-thermal unit commitment". In: *Mathematical Methods of Operations Research* 80.3 (2014), pp. 227–253.

[3] W. van Ackooij. "Eventual Convexity of Chance Constrained Feasible Sets". In: *Optimization (A Journal of Math. Programming and Operations Research)* 64.5 (2015), pp. 1263–1284.

[4] W. van Ackooij, V. Berge, W. de Oliveira, and C. Sagastizábal. "Probabilistic optimization via approximate p-efficient points and bundle methods". In: *Computers & Operations Research* 77 (2017), pp. 177 –193. ISSN: 0305-0548.

[5] W. van Ackooij, J. B. Cruz, and W. de Oliveira. "A strongly convergent proximal bundle method for convex minimization in Hilbert spaces". In: *Optimization* 65.1 (2016), pp. 145–167.

[6] W. van Ackooij and R. Henrion. "Gradient Formulae for nonlinear probabilistic constraints with Gaussian and Gaussian-Like Distributions". In: *SIAM Journal on Optimization* 24.4 (2014), pp. 1864–1889.

[7] W. van Ackooij, R. Henrion, A. Möller, and R. Zorgati. "Joint Chance Constrained Programming for Hydro Reservoir Management". In: *Optimization and Engineering* 15 (2014), pp. 509–531.

[8] W. van Ackooij, R. Henrion, A. Möller, and R. Zorgati. "On probabilistic constraints induced by rectangular sets and multivariate normal distributions". In: *Mathematical Methods of Operations Research* 71.3 (2010), pp. 535–549.

[9] W. van Ackooij and W. de Oliveira. "Level Bundle Methods for Constrained Convex Optimization with Various Oracles". In: *Computational Optimization and Applications* 57.3 (2014), pp. 555–597.

[10] W. van Ackooij and C. Sagastizábal. "Constrained Bundle Methods for Upper Inexact Oracles with Application to Joint Chance Constrained Energy Problems". In: *SIAM Journal on Optimization* 24.2 (2014), pp. 733–765.

[11] W. van Ackooij and W. de Oliveira. "Convexity and optimization with copulæstructured probabilistic constraints". In: *Optimization* 65.7 (2016), pp. 1349–1376.

[12] W. van Ackooij and A. Frangioni. "Incremental Bundle Methods using Upper Models". In: *SIAM Journal on Optimization* 28.1 (2018), pp. 379–410.

[13] M. V. Afonso, J. M. Bioucas-Dias, and M. A. T. Figueiredo. "Fast Image Recovery Using Variable Splitting and Constrained Optimization". In: *IEEE Transactions on Image Processing* 19.9 (2010), pp. 2345–2356.

[14] F. Alvarez. "Weak Convergence of a Relaxed and Inertial Hybrid Projection-Proximal Point Algorithm for Maximal Monotone Operators in Hilbert Space". In: *SIAM Journal on Optimization* 14.3 (2004), pp. 773–782.

[15] F. Alvarez and H. Attouch. "An Inertial Proximal Method for Maximal Monotone Operators via Discretization of a Nonlinear Oscillator with Damping". In: *Set-Valued Analysis* 9.1 (2001), pp. 3–11.

[16] L. T. H. An and P. D. Tao. "The DC (Difference of Convex Functions) Programming and DCA Revisited with DC Models of Real World Nonconvex Optimization Problems". In: *Annals of Operations Research* 133.1 (2005), pp. 23–46.

[17] T. Arnold, R. Henrion, A. Möller, and S. Vigerske. "A mixed-integer stochastic nonlinear optimization problem with joint probabilistic constraints". In: *Pacific Journal of Optimization* 10 (2014), pp. 5–20.

[18] T. Asamov and W. B. Powell. "Regularized Decomposition of High-Dimensional Multistage Stochastic Programs with Markov Uncertainty". In: *SIAM Journal on Optimization* 28.1 (2018), pp. 575–595.

[19] A. M. Bagirov and J. Yearwood. "A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems". In: *European Journal of Operational Research* 170.2 (2006), pp. 578 –596.

[20] A. Beck and M. Teboulle. "Fast Gradient-Based Algorithms for Constrained Total Variation Image Denoising and Deblurring Problems". In: *IEEE Transactions on Image Processing* 18.11 (2009), pp. 2419–2434.

[21] J. Y. Bello-Cruz and W. de Oliveria. "Level bundle-like algorithms for convex optimization". In: *Journal of Global Optimization* 59.4 (2014), pp. 787–809.

[22] J. Y. Bello Cruz. "On Proximal Subgradient Splitting Method for Minimizing the sum of two Nonsmooth Convex Functions". In: *Set-Valued and Variational Analysis* 25.2 (2017), pp. 245–263.

[23] F. Beltran, W. de Oliveira, and E. C. Finardi. "Application of Scenario Tree Reduction Via Quadratic Process to Medium-Term Hydrothermal Scheduling Problem". In: *IEEE Transactions on Power Systems* 32.6 (2017), pp. 4351–4361.

[24] H. Ben Amor, J. Desrosiers, and A. Frangioni. "On the Choice of Explicit Stabilizing Terms in Column Generation". In: *Discrete Applied Mathematics* 157.6 (2009), pp. 1167–1184.

[25] A. Ben-Tal and A. Nemirovski. "Non-euclidean restricted memory level method for large-scale convex optimization". In: *Math. Program.* 102 (3 2005), pp. 407–456. ISSN: 0025-5610.

[26] J. Benders. "Partitioning procedures for solving mixed-variables programming problems". In: *Numerische Mathematik* 4.1 (1962), pp. 238–252.

[27] J. R. Birge. "Decomposition and Partitioning Methods for Multistage Stochastic Linear Programs". In: *Operations Research* 33.5 (1985), pp. 989–1007.

[28] J. Birge and F. Louveaux. *Introduction to Stochastic Programming*. Springer, New York, 1997.

[29] J. Bonnans, J. Gilbert, C. Lemaréchal, and C. Sagastizábal. *Numerical Optimization: Theoretical and Practical Aspects*. 2nd. Springer-Verlag, 2006, p. 490.

[30] U. Brannlund, K. C. Kiwiel, and P. O. Lindberg. "A descent proximal level bundle method for convex nondifferentiable optimization". In: *Operations Research Letters* 17.3 (1995), pp. 121 –126.

[31] I. Bremer, R. Henrion, and A. Möller. "Probabilistic constraints via SQP solver: Application to a renewable energy management problem." In: *Computational Management Science* 12 (2015), pp. 435–459.

[32] S. V. Bruno, L. A. Moraes, and W. de Oliveira. "Optimization techniques for the Brazilian natural gas network planning problem". In: *Energy Systems (ENSY)* 8.1 (2017), pp. 81–101.

[33] G. C. Calafiore and M. C. Campi. "Uncertain Convex Programs: Randomized Solutions and Confidence Levels". In: *Mathematical Programming* 102.1 (2005), pp. 25–46.

[34] M. C. Campi and S. Garatti. "A Sampling-and-Discarding Approach to Chance-Constrained Optimization: Feasibility and Optimality". In: *Journal of Optimization Theory and Applications* 148.2 (2011), pp. 257–280.

[35] C. C. Caroe and J. Tind. "L-shaped decomposition of two-stage stochastic programs with integer recourse". In: *Math. Programming* 83 (1998), pp. 451–464.

[36] V. Caselles et al. "An introduction to Total Variation for Image Analysis". In: *Theoretical Foundations and Numerical Methods for Sparse Recovery, De Gruyter, Radon Series Comp. Appl. Math.* 9 (2010), pp. 263–340.

[37] E. Castillo et al. "Estimating the parameters of a fatigue model using Benders decomposition". In: *Annals of Operations Research* 210.1 (2013), 309331.

[38] Y. Chen, G. Lan, Y. Ouyang, and W. Zhang. *Fast Bundle-Level Type Methods for unconstrained and ball-constrained convex optimization.* Tech. rep. 1412.2128. 2014, p. 29.

[39] J. Cheng, M. Houda, and A. Lisser. *Second-order cone programming approach for elliptically distributed joint probabilistic constraints with dependent rows.* Tech. rep. 2014, pp. 1–26.

[40] G. Codato and M. Fischetti. "Combinatorial Benders' Cuts for Mixed-Integer Linear Programming." In: *Operations Research* 54.4 (2006), pp. 756–766.

[41] L. Condat. "A Primal–Dual Splitting Method for Convex Optimization Involving Lipschitzian, Proximable and Linear Composite Terms". In: *Journal of Optimization Theory and Applications* 158.2 (2013), pp. 460–479.

[42] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction To Derivative-Free Optimization.* Society For Industrial And Applied Mathematics, 2009, pp. 1–289.

[43] R. Correa and C. Lemaréchal. "Convergence of some algorithms for convex minimization". In: *Mathematical Programming* 62.2 (1993), pp. 261–275.

[44] A. M. Costa. "A survey on benders decomposition applied to fixed-charge network design problems". In: *Computers & Operations Research* 32.6 (2005), 14291450.

[45] J. Y. B. Cruz and W. de Oliveira. "On Weak and Strong Convergence of the Projected Gradient Method for Convex Optimization in Real Hilbert Spaces". In: *Numerical Functional Analysis and Optimization* 37.2 (2016), pp. 129–144.

[46] J. Currie and D. I. Wilson. "OPTI: Lowering the Barrier Between Open Source Optimizers and the Industrial MATLAB User". In: *Foundations of Computer-Aided Process Operations.* Ed. by N. Sahinidis and J. Pinto. Savannah, Georgia, USA, 2012.

[47] W. de Oliveira. *Proximal bundle methods for nonsmooth DC programming.* Tech. rep. Available at `www.oliveira.mat.br`., 2017, p. 25.

[48] W. de Oliveira and M. Solodov. "A doubly stabilized bundle method for nonsmooth convex optimization". In: *Mathematical Programming* 156.1 (2016), pp. 125–159.

[49] W. de Oliveira. "Combining level and proximal bundle methods for convex optimization in energy problems." In: *EngOpt 2012 - International Conference on Engineering Optimization.* 404. 2012, pp. 1–10.

[50] W. de Oliveira. "Target radius methods for nonsmooth convex optimization". In: *Operations Research Letters* 45.6 (2017), pp. 659–664.

[51] W. de Oliveira and M. P. Tcheou. "An Inertial Algorithm for DC Programming". In: *Set-Valued and Variational Analysis* (2018).

[52] W. L. de Oliveira et al. "Optimal scenario tree reduction for stochastic streamflows in power generation planning problems". In: *Optimization Methods and Software* 25.6 (2010), pp. 917–936.

[53] A. Delfino and W. de Oliveira. "Outer-approximation algorithms for nonsmooth convex MINLP problems". In: *Optimization* 67.6 (2018), pp. 797–819.

[54] D. Dentcheva. "Optimisation Models with Probabilistic Constraints." In: *Lectures on Stochastic Programming. Modeling and Theory.* Ed. by A. Shapiro, D. Dentcheva, and A. Ruszczyński. Vol. 9. MPS-SIAM series on optimization. SIAM and MPS, Philadelphia, 2009, pp. 87–154.

[55] D. Dentcheva, B. Lai, and A. Ruszczyński. "Dual methods for probabilistic optimization problems". In: *Mathematical Methods of Operations Research* 60.2 (2004), pp. 331–346.

[56] D. Dentcheva and G. Martinez. "Regularization methods for optimization problems with probabilistic constraints". In: *Math. Programming (series A)* 138.1-2 (2013), pp. 223–251.

[57] D. Dentcheva and G. Martinez. "Two-stage stochastic optimization problems with stochastic ordering constraints on the recourse". In: *European Journal of Operational Research* 219.1 (2012), pp. 1–8.

[58] D. Dentcheva, A. Prékopa, and A. Ruszczyński. "Concavity and efficient points for discrete distributions in stochastic programming". In: *Mathematical Programming* 89 (2000), pp. 55–77.

[59] D. Dentcheva and G. Martinez. "Augmented Lagrangian method for probabilistic optimization". In: *Annals of Operations Research* 200.1 (2011), pp. 109–130.

[60] J. V. Dinter et al. "The unit commitment model with concave emissions costs: a hybrid Benders Decomposition with nonconvex master problems". In: *Annals of Operations Research* 210.1 (2013), pp. 361–386.

[61] E. D. Dolan and J. J. Moré. "Benchmarking optimization software with performance profiles". In: *Mathematical Programming* 91 (2 2002), pp. 201–213.

[62] J. Dupačová, N. Gröwe-Kuska, and W. Römisch. "Scenario reduction in stochastic programming : An approach using probability metrics". In: *Mathematical Programming* 95.3 (2003), pp. 493–511.

[63] M. Duran and I. E. Grossmann. "An outer-approximation algorithm for a class of mixed-integer nonlinear programs". English. In: *Mathematical Programming* 36.3 (1986), pp. 307–339. ISSN: 0025-5610.

[64] J. Elzinga and T. G. Moore. "A central cutting plane method for the convex programming problem". In: *Mathematical Programming* 8 (1975), pp. 134–145.

[65] C. Fábián. "Bundle-type methods for inexact data". In: *Proceedings of the XXIV Hungarian Operations Researc Conference (Veszprém, 1999)*. Vol. 8 (special issue, T. Csendes and T. Rapcsk, eds.) 2000, pp. 35–55.

[66] C. Fábián and Z. Szőke. "Solving Two-Stage Stochastic Programming Problems with Level Decomposition". In: *Computational Management Science* 4 (2007), pp. 313–353.

[67] A. Fakhri and M. Ghatee. "Solution of preemptive multi-objective network design problems applying Benders decomposition method". In: *Annals of Operations Research* 210.1 (2013), pp. 295–307.

[68] R. Fletcher and S. Leyffer. "Solving mixed integer nonlinear programs by outer approximation". In: *Mathematical Programming* 66.1-3 (1994), pp. 327–349.

[69] C. A. Floudas. *Generalized Benders Decomposition. Appearing in [70].* 2nd. Springer - Verlag, 2009, pp. 1163–1174.

[70] C. Floudas and P. P. (Eds). *Encyclopedia of Optimization.* 2nd. Springer - Verlag, 2009, pp. 1–4067.

[71] A. Frangioni and B. Gendron. "A Stabilized Structured Dantzig-Wolfe Decomposition Method". In: *Mathematical Programming B* 104.1 (2013), pp. 45–76.

[72] A. Frangioni and E. Gorgone. "Generalized Bundle Methods for Sum-Functions with "Easy" Components: Applications to Multicommodity Network Design". In: *Mathematical Programming* 145.1 (2014), pp. 133–161.

[73] M. Gaudioso, G. Giallombardo, G. Miglionico, and A. M. Bagirov. "Minimizing nonsmooth DC functions via successive DC piecewise-affine approximations". In: *Journal of Global Optimization* 71 (1 2018), pp. 37–55.

[74] M. Gaudioso, G. Giallombardo, and G. Miglionico. "Minimizing Piecewise-Concave Functions Over Polyhedra". In: *Mathematics of Operations Research* 43.2 (2018), pp. 580–597.

[75] A. Genz and F. Bretz. *Computation of multivariate normal and t probabilities.* Lecture Notes in Statistics 195. Springer, Dordrecht, 2009.

[76] A. M. Geoffrion. "Generalized Benders Decomposition". In: *Journal of Optimization Theory and Applications* 10.4 (1972), pp. 237–260.

[77] M. Ghotboddini, M. Rabbani, and H. Rahimian. "A comprehensive dynamic cell formation design: Benders decomposition approach". In: *Expert Systems with Applications* 38.3 (2011), 24782488.

[78] R. C. Gonzalez and R. E. Woods. *Digital Image Processing (3rd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2006. ISBN: 013168728X.

[79] T. G. Grandón, H. Heitsch, and R. Henrion. "A joint model of probabilistic/robust constraints for gas transport management in stationary networks". In: *Computational Management Science* 14.3 (2017), pp. 443–460.

[80] N. Gröwe-Kuska, H. Heitsch, and W. Römisch. "Scenario Reduction and Scenario Tree Construction for Power Management Problems". In: *IEEE Bologna Power Tech. Conference* (2003), p. 7.

[81] T. Gruzdeva and A. Strekalovsky. "A D.C. Programming Approach to Fractional Problems". In: *Learning and Intelligent Optimization: 11th International Conference, LION 11, Nizhny Novgorod, Russia, June 19-21, 2017, Revised Selected Papers*. Ed. by R. Battiti, D. E. Kvasov, and Y. D. Sergeyev. Cham: Springer International Publishing, 2017, pp. 331–337.

[82] H. Heitsch and W. Römisch. "Scenario Reduction Algorithms in Stochastic Programming". In: *Computation Optimization and Applications* 24.2-3 (2003), pp. 187–206.

[83] H. Heitsch and W. Römisch. "Scenario Tree Reduction for Multistage Stochastic Programs". In: *Computational Management Science* 6 (2009), pp. 117–133.

[84] L. Hellemo, K. Midthun, A. Tomasgard, and A. Werner. "Multi-Stage Stochastic Programming for Natural Gas Infrastructure Design with a Production Perspective". In: *Stochastic Programming*. 2013. Chap. 10, pp. 259–288.

[85] L. Hellemo, K. Midthun, A. Tomasgard, and A. Werner. "Natural Gas Infrastructure Design with an Operational Perspective". In: *Energy Procedia* 26.0 (2012). 2nd Trondheim Gas Technology Conference, pp. 67 –73. ISSN: 1876-6102.

[86] R. Henrion. "Gradient estimates for Gaussian distribution functions: Application to probabilistically constrained optimization problems". In: *Numerical Algebra, Control and Optimization* 2 (2012), pp. 655–668.

[87] R. Henrion. "Introduction to chance constraint programming". In: *Tutorial paper for the Stochastic Programming Community HomePage, http://www.wias-berlin.de/people/henrion/publikat.html* (2004).

[88] R. Henrion and A. Möller. "Optimization of a continuous distillation process under random inflow rate". In: *Computer & Mathematics with Applications* 45 (2003), pp. 247–262.

[89] R. Henrion and W. Römisch. "Metric regularity and quantitative stability in stochastic programs with probabilistic constraints." In: *Mathematical Programming* 84 (1999), pp. 55–88.

[90] R. Henrion and C. Strugarek. "Convexity of Chance Constraints with Dependent Random Variables: the use of Copulae." In: *Stochastic Optimization Methods in Finance and Energy: New Financial Products and Energy Market Strategies*. Ed. by M. Bertocchi, G. Consigli, and M. Dempster. International Series in Operations Research and Management Science. Springer, 2011, pp. 427–439.

[91] R. Henrion and C. Strugarek. "Convexity of Chance Constraints with Independent Random Variables". In: *Computational Optimization and Applications* 41 (2008), pp. 263–276.

[92] M. Hintermüller. "A Proximal Bundle Method Based on Approximate Subgradients". In: *Computational Optimization and Applications* 20 (3 2001). 10.1023/A:1011259017643, pp. 245–266.

[93] J.-B. Hiriart-Urruty. "Generalized Differentiability / Duality and Optimization for Problems Dealing with Differences of Convex Functions". In: *Convexity and Duality in Optimization: Proceedings of the Symposium on Convexity and Duality in Optimization Held at the University of Groningen, The Netherlands June 22, 1984*. Ed. by J. Ponstein. Berlin, Heidelberg: Springer Berlin Heidelberg, 1985, pp. 37–70.

[94] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms*. Grund. der math. Wiss 305-306. (two volumes). Springer-Verlag, 1993.

[95] J. Hiriart-Urruty. *Optimisation et analyse convexe*. Presses Universitaires de France, 1998. ISBN: 9782130489832.

[96] J. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms I*. 2nd. Grundlehren der mathematischen Wissenschaften 305. Springer-Verlag, 1996, p. 418.

[97] K. Holmberg and H. Tuy. "A production-transportation problem with stochastic demand and concave production costs". In: *Mathematical Programming* 85.1 (1999), pp. 157–179.

[98] J. N. Hooker and G. Ottosson. "Logic-based Benders decomposition". In: *Math. Programming* 96 (2003), pp. 33–60.

[99] M. Houda and A. Lisser. "On the Use of Copulas in Joint Chance-constrained Programming". In: *Proceedings of the 3rd International Conference on Operations Research and Enterprise Systems*. 2014, pp. 72–79.

[100] A. F. Izmailov and M. V. Solodov. *Newton-Type Methods for Optimization and Variational Problems*. 1st. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, 2014, p. 573.

[101] K. Joki, A. M. Bagirov, N. Karmitsa, and M. M. Mäkelä. "A proximal bundle method for nonsmooth DC optimization utilizing nonconvex cutting planes". In: *Journal of Global Optimization* 68.3 (2017), pp. 501–535.

[102] S. Kazempour and A. Conejo. "Strategic Generation Investment Under Uncertainty Via Benders Decomposition". In: *Power Systems, IEEE Transactions on* 27.1 (2012), pp. 424–432.

[103] J. Kelley. "The Cutting-Plane Method for Solving Convex Programs". In: *Journal of the Society for Industrial and Applied Mathematics* 8.4 (1960), pp. 703–712.

[104] W. Khalaf, A. Astorino, P. d'Alessandro, and M. Gaudioso. "A DC optimization-based clustering technique for edge detection". In: *Optimization Letters* 11.3 (2017), pp. 627–640.

[105] K. Kiwiel. "A proximal bundle method with approximate subgradient linearizations". In: *SIAM Journal on Optimization* 16.4 (2006), pp. 1007–1023.

[106] K. Kiwiel. "Proximal level bundle methods for convex nondifferentiable optimization, saddle-point problems and variational inequalities." In: *Math. Programming* 69.1 (1995), pp. 89–109.

[107] A. J. Kleywegt, A. Shapiro, and T. Homem-de Mello. "The Sample Average Approximation Method for Stochastic Discrete Optimization". In: *SIAM Journal on Optimization* 12.2 (Feb. 2002), pp. 479–502. ISSN: 1052-6234.

[108] A. Kolokolov and N. Kosarev. "Analysis of Decomposition Algorithms with Benders Cuts for p-Median Problem". In: *Journal of Mathematical Modelling and Algorithms* 5.2 (2006), pp. 189–199.

[109] R. M. Kovacevic and A. Pichler. "Tree approximation for discrete time stochastic processes: a process distance approach". In: *Annals of Operations Research* 235.1 (2015), pp. 395–421.

[110] G. Lan. "Bundle-level type methods uniformly optimal for smooth and nonsmooth convex optimization". In: *Mathematical Programming* 149.1 (2015), pp. 1–45.

[111] A. Lanza, S. Morigi, and F. Sgallari. "Convex Image Denoising via Non-convex Regularization with Parameter Selection". In: *Journal of Mathematical Imaging and Vision* 56.2 (2016), pp. 195–220. ISSN: 1573-7683.

[112] H. A. Le Thi and T. Pham Dinh. "DC programming in communication systems: challenging problems and methods". In: *Vietnam Journal of Computer Science* 1.1 (2014), pp. 15–28. ISSN: 2196-8896.

[113] H. A. Le Thi, T. Pham Dinh, and H. V. Ngai. "Exact penalty and error bounds in DC programming". In: *Journal of Global Optimization* 52.3 (2012), pp. 509–535.

[114] M. A. Lejeune. "Pattern-Based Modeling and Solution of Probabilistically Constrained Optimization Problems". In: *Operations Research* 60.6 (2012), pp. 1356–1372.

[115] M. A. Lejeune. "Pattern definition of the p-efficiency concept". In: *Annals of Operations Research* 200 (2012), pp. 23–36.

[116] M. A. Lejeune and N. Noyan. "Mathematical Programming Approaches for Generating p-Efficient Points". In: *European Journal of Operational Research* 207.2 (2010), pp. 590–600.

[117] M. A. Lejeune and A. Ruszczyński. "An efficient Trajectory Method for Probablistic Production-Inventory-Distribution Problems". In: *Operations Research* 55.2 (2007), pp. 378–394.

[118] M. A. Lejeune and F. Margot. "Solving Chance-Constrained Optimization Problems with Stochastic Quadratic Inequalities". In: *Operations Research* 64.4 (2016), pp. 939–957.

[119] C. Lemaréchal. "Lagrangian decomposition and nonsmooth optimization: bundle algorithm, prox iteration, augmented lagrangian". In: *Nonsmooth optimization methods and applications*. Ed. by F. Giannessi. Gordon & Breach, 1992, pp. 201–216.

[120] C. Lemaréchal, J. Malick, and S. Zaourar. "Coûts Marginaux en Production Journalière". In: *Rapport Final du Contrat* (2011), pp. 1–7.

[121] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. "New variants of bundle methods". In: *Math. Programming* 69.1 (1995), pp. 111–147.

[122] C. Lemaréchal and A. Renaud. "A geometric study of duality gaps, with applications". In: *Math. Programming* 90 (2001), pp. 399–427.

[123] C. Lemaréchal, A. Nemirovskii, and Y. Nesterov. "New variants of Bundle Methods". In: *Math. Program.* 69.1 (1995), pp. 111–147. ISSN: 0025-5610.

[124] X. Li, E. Armagan, A. Tomasgard, and P. I. Barton. "Long-term planning of natural gas production systems via a stochastic pooling problem". In: *American Control Conference (ACC), 2010*. 2010, pp. 429 –435.

[125] X. Li, Y. Chen, and P. I. Barton. "Nonconvex Generalized Benders Decomposition with Piecewise Convex Relaxations for Global Optimization of Integrated Process Design and Operation Problems". In: *Industrial & Engineering Chemistry Research* 51.21 (2012), pp. 7287–7299.

[126] C. w. Lu. "Image restoration and decomposition using nonconvex non-smooth regularisation and negative Hilbert-Sobolev norm". In: *IET Image Processing* 6.6 (2012), pp. 706–716.

[127] J. Luedtke. "A branch-and-cut decomposition algorithm for solving chance-constrained mathematical programs with finite support". In: *Mathematical Programming* 146.1-2 (2014), pp. 219–244.

[128] J. Luedtke and S. Ahmed. "A Sample Approximation Approach for Optimization with Probabilistic Constraints". In: *SIAM Journal on Optimization* 19 (2008), pp. 674–699.

[129] J. Luedtke, S. Ahmed, and G. Nemhauser. "An integer programming approach for linear programs with probabilistic constraints". In: *Mathematical Programming* 122.2 (2010), pp. 247–272.

[130] M. E. P. Maceira et al. "Ten years of application of stochastic dual dynamic programming in official and agent studies in Brazil – description of the NEWAVE program". In: *Power Systems Computation Conference, 2008*. 2008, pp. 429 –435.

[131] T. L. Magnanti and R. T. Wong. "Accelerating Benders decomposition: algorithmic enhancement and model selection criteria." In: *Operations Research* 29.3 (1981), pp. 464–484.

[132] P. Mahey, A. Benchakroun, and F. Boyer. "Capacity and flow assignment of data networks by generalized Benders decomposition". In: *Journal of Global Optimization* 20.2 (2001), pp. 173–193.

[133] J. Malick, W. de Oliveira, and S. Zaourar. "Uncontrolled inexact information within bundle methods". In: *EURO Journal on Computational Optimization* 5.1 (2017), pp. 5–29.

[134] J. Mayer. *On the Numerical solution of jointly chance constrained problems. Chapter 12 in [197]*. 1st. Springer, 2000, pp. 220–235.

[135] A. McNeil and J. Nešlehová. "Multivariate Archimedian Copulas, D-Monotone functions and l_1 norm symmetric distributions". In: *The Annals of Statistics* 37 (2009), pp. 3059–3097.

[136] R. Montemanni and L. M. Gambardella. "The robust shortest path problem with interval data via Benders decomposition". In: *4OR* 3.4 (2005), 315328.

[137] J. J. Moreau. "Proximité et dualité dans un espace Hilbertien". In: *Bulletin de la Société Mathématique de France* 93 (1965), pp. 273–299.

[138] D. Morgan, J. Eheart, and A. Valocchi. "Aquifer remediation design under uncertainty using a new chance constraint programming technique". In: *Water Resources Research* 29 (1993), pp. 551–561.

[139] A. Moudafi and M. Oliny. "Convergence of a splitting inertial proximal method for monotone operators". In: *Journal of Computational and Applied Mathematics* 155.2 (2003), pp. 447 –454. ISSN: 0377-0427.

[140] E. Munõz and M. Stolpe. "Generalized Benders Decomposition for topology optimization problems". In: *Journal of Global Optimization* 51.1 (2011), pp. 149–183.

[141] R. B. Nelsen. *An Introduction to Copulas*. 2nd. Springer Series in Statistics. Springer, 2006, p. 272.

[142] Y. Nesterov. *Introductory Lectures on Convex Optimization. A Basic Course*. Vol. 87. Applied Optimization. Springer Science, 2004, pp. 1–236.

[143] N. Newhan. "Power system investment planning using stochastic dual dynamic programming". http://hdl.handle.net/10092/1975. PhD thesis. University of Canterbury, New Zealand, 2008.

[144] M. Nikolova, M. K. Ng, and C. P. Tam. "Fast Nonconvex Nonsmooth Minimization Methods for Image Restoration and Reconstruction". In: *IEEE Transactions on Image Processing* 19.12 (2010), pp. 3073–3088.

[145] "Nonconvex Generalized Benders Decomposition for Stochastic Separable Mixed-Integer Nonlinear Programs". In: *Journal of Optimization Theory and Applications* 151 (3 2011), pp. 425–454. ISSN: 0022-3239.

[146] P. Ochs, Y. Chen, T. Brox, and T. Pock. "iPiano: Inertial Proximal Algorithm for Nonconvex Optimization". In: *SIAM Journal on Imaging Sciences* 7.2 (2014), pp. 1388–1419.

[147] W. de Oliveira and C. Sagastizábal. "Bundle methods in the XXI century: A birds'-eye view". In: *Pesquisa Operaciona* 34.3 (2014), pp. 647–670.

[148] W. de Oliveira and C. Sagastizábal. "Level Bundle Methods for Oracles with On Demand Accuracy". In: *Optimization Methods and Software* 29.6 (2014), pp. 1180–1209.

[149] W. de Oliveira, C. Sagastizábal, and C. Lemaréchal. "Convex proximal bundle methods in depth: a unified analysis for inexact oracles". In: *Math. Prog. Series B* 148 (2014), pp. 241–277.

[150] W. de Oliveira. "Regularized optimization methods for convex MINLP problems". In: *TOP* 24.3 (2016), pp. 665–692.

[151] H. Osman and K. Demirli. "A bilinear goal programming model and a modified Benders decomposition algorithm for supply chain reconfiguration and supplier selection". In: *Int. J. Production Economics* 124 (2010), pp. 97–105.

[152] N. Oudjane. *Utilisation des copules pour la gestion du risque*. Tech. rep. HI-23/2002/006. EDF R&D, 2002, pp. 1–50.

[153] A. Ouorou. "A proximal cutting plane method using Chebychev center for nonsmooth convex optimization". In: *Math. Program.* 119.2 (2009), pp. 239–271.

[154] G. Pagès and J. Printems. "Optimal quadratic quantization for numerics: the Gaussian case". In: *Monte Carlo Methods and Applications* 9.2 (2003), pp. 135–166.

[155] J.-S. Pang, M. Razaviyayn, and A. Alvarado. "Computing B-Stationary Points of Nonsmooth DC Programs". In: *Mathematics of Operations Research* 42.1 (2017), pp. 95–118.

[156] B. S. Pay and Y. Song. "Partition-based decomposition algorithms for two-stage Stochastic integer programs with continuous recourse". In: *Annals of Operations Research* (2017).

[157] M. Pereira and L. Pinto. "Multi-stage Stochastic optimization applied to energy planning". In: *Mathematical Programming* 52.2 (1991), pp. 359–375.

[158] G. C. Pflug and A. Pichler. "A Distance For Multistage Stochastic Optimization Models". In: *SIAM Journal on Optimization* 22.1 (2012), pp. 1–23.

[159] G. C. Pflug and A. Pichler. "Approximations for Probability Distributions and Stochastic Optimization Problems". In: *Stochastic Optimization Methods in Finance and Energy*. Springer New York, 2011, pp. 343–387.

[160] G. C. Pflug and M. Pohl. "A Review on Ambiguity in Stochastic Portfolio Optimization". In: *Set-Valued and Variational Analysis* (2017).

[161] G. C. Pflug and W. Römisch. *Modeling, Measuring and Managing Risk*. WORLD SCIENTIFIC, 2007, p. 304.

[162] B. Polyak. "Some methods of speeding up the convergence of iteration methods". In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17.

[163] A. Prékopa. "Dual Method for a one-stage stochastic programming problem with random rhs obeying a discrete probabiltiy distribution." In: *Z. Operations Research* 34 (1990), pp. 441–461.

[164] A. Prékopa. "Logarithmic Concave Measures with Applications to Stochastic Programming". In: *Acta Scientiarium Mathematicarum (Szeged)* 32 (1971), pp. 301–316.

[165] A. Prékopa. "Probabilistic programming." In: *Stochastic Programming*. Ed. by A. Ruszczyński and A. Shapiro. Vol. 10. Handbooks in Operations Research and Management Science. Elsevier, Amsterdam, 2003, pp. 267–351.

[166] A. Prékopa. *Stochastic Programming*. Kluwer, Dordrecht, 1995.

[167] A. Prékopa and T. Szántai. "Flood Control reservoir system design using stochastic programming". In: *Math. Programming Study* 9 (1978), pp. 138–151.

[168] A. Prékopa and T. Szántai. "On optimal regulation of a storage level with application to the water level regulation of a lake". In: *European Journal of Operations Research* 3 (1979), pp. 175–189.

[169] A. Prékopa, B. Vízvári, and T. Badics. "Programming under probabilistic constraints with discrete random variable." In: *New Trends in Mathematical Programming : Hommage to Steven Vajda*. Ed. by F. Giannessi, S. Komlósi, and T. Rapcsák. Vol. 13. Applied Optimization. Springer, 1998, pp. 235–255.

[170] S. Rebenack. "Combining sampling-based and scenario-based nested Benders decomposition methods: application to stochastic dual dynamic programming". In: *Math. Programming* To Appear (2015), pp. 1–47.

[171] R. T. Rockafellar. "Monotone operators and the proximal point algorithm". In: *SIAM Journal on Control and Optimization* 14 (1976), pp. 877–898.

[172] R. T. Rockafellar and R.-B. Wets. "A Lagrangian finite generation technique for solving linear-quadratic problems in stochastic programming". In: *Mathematical Programming Study* 28 (1986), pp. 63–93.

[173] R. Rockafellar. *Convex Analysis*. 1st. Princeton University Press, 1970, p. 472.

[174] A Ruszczyński. "A Regularized Decomposition Method for Minimizing a Sum of Polyhedral Functions". In: *Math. Program.* 35 (3 1986), pp. 309–333.

[175] A. Ruszczyński. "Probabilistic programming with discrete distributions and precedence constrained knapsack polyhedra." In: *Mathematical Programming* 93 (2002), pp. 195–215.

[176] C. Sagastizábal. "Divide to conquer: decomposition methods for energy optimization". In: *Mathematical Programming* 134 (1 2012), pp. 187–222. ISSN: 0025-5610.

[177] G. K. D. Sahiridis, M. G. Ierapetritou, and C. A. Floudas. *Benders Decomposition and Its Application in Engineering*. Vol. 210. Annals of Operations Research. Springer, 2013, pp. 1–432.

[178] G. K. D. Sahiridis, M. Minoux, and M. G. Ierapetritou. "Accelerating Benders method using covering cut bundle generation". In: *International Transactions In Operational Research* 17 (2010), pp. 221–237.

[179] T. Santoso, S. Ahmed, M. Goetschalckx, and A. Shapiro. "A stochastic programming approach for supply chain network design under uncertainty". In: *European Journal of Operational Research* 167.1 (2005), pp. 96 –115. ISSN: 0377-2217.

[180] P. Schütz, A. Tomasgard, and S. Ahmed. "Supply chain design under uncertainty using sample average approximation and dual decomposition". In: *European Journal of Operational Research* 199.2 (2009), pp. 409 –419.

[181] A. Shapiro, D. Dentcheva, and A. Ruszczyński. *Lectures on Stochastic Programming. Modeling and Theory*. Vol. 9. MPS-SIAM series on optimization. SIAM and MPS, Philadelphia, 2009.

[182] H. Sherali and B. J. Lunday. "On generating maximal nondominated Benders cuts". In: *Annals of Operations Research* 210.1 (2013), pp. 57–72.

[183] K. Singh, A. B. Philpott, and R. Wood. "Dantzig-Wolfe Decomposition for Solving Multistage Stochastic Capacity-Planning Problems". In: *Operations Research* 57.5 (2009), pp. 1271–1286.

[184] A. Sklar. "Fonctions de répartition à dimensions et leurs marges". In: *Publications and l'Institut de Statistique de Paris* 8 (1959), pp. 229–231.

[185] A. Sklar. "Random variables, joint distribution functions, and copulas". In: *Kybernetika* 9.6 (1973), pp. 449–460.

[186] R. van Slyke and R.-B. Wets. "L-shaped linear programs with applications to optimal control and stochastic programming". In: *SIAM Journal of Applied Mathematics* 17 (1969), pp. 638–663.

[187] M. V. Solodov. "A Bundle Method for a Class of Bilevel Nonsmooth Convex Minimization Problems". In: *SIAM Journal on Optimization* 18.1 (2007), pp. 242–259.

[188] J. C. O. Souza, P. R. Oliveira, and A. Soubeyran. "Global convergence of a proximal linearized algorithm for difference of convex functions". In: *Optimization Letters* 10.7 (2016), pp. 1529–1539.

[189] A. Strekalovskiy. "An exact penalty method for D.C. optimization". In: *AIP Conference Proceedings* 1776.1 (2016), p. 060003.

[190] A. Strekalovsky and I. Minarchenko. "On local search in D.C. optimization". In: *2017 Constructive Nonsmooth Analysis and Related Topics (dedicated to the memory of V.F. Demyanov) (CNSA)*. 2017, pp. 1–4.

[191] A. S. Strekalovsky. "On local search in d.c. optimization problems". In: *Applied Mathematics and Computation* 255.1 (2015), pp. 73–83.

[192] E. Tamm. "On g-concave functions and Probability measures (Russian)". In: *Eesti NSV Teaduste Akademia Toimetised, Füüsika-Matemaatika* 28 (1977), pp. 17–24.

[193] P. D. Tao and L. T. H. An. "Convex analysis approach to DC programming: theory, algorithms and applications". In: *Acta Mathematica Vietnamica* 22.1 (1997), pp. 289–355.

[194] P. D. Tao and E. B. Souad. "Algorithms for Solving a Class of Nonconvex Optimization Problems. Methods of Subgradients". In: *North-Holland Mathematics Studies* 129 (1986), pp. 249 –271.

[195] P. K. Trivedi and D. M. Zimmer. "Copula Modeling: An Introduction for Practitioners". In: *Foundations and Trends in Econometrics* 1.1 (2007), pp. 1–111.

[196] H. Tuy. *Convex Analysis and Global Optimization*. 2nd. Springer Optimization and Its Applications. Springer, 2016, pp. XVI, 505. ISBN: 1931-6828.

[197] S. Uryas'ev (ed). *Probabilistic Constrained Optimization: Methodology and Applications*. Kluwer Academic Publishers, 2000, pp. 1–320.

[198] H. Üster, G. Easwaran, E. Akçali, and S. Çetinkaya. "Benders decomposition with alternative multiple cuts for a multi-product closed-loop supply chain network design model". In: *Naval Research Logistics (NRL)* 54.8 (2007), pp. 890–907.

[199] W. van Ackoij, W. de Oliveira, and Y. Song. *On regularization with normal solutions in decomposition methods for multistage stochastic programming*. Tech. rep. 5806. 2017, pp. 1–27. URL: `http://www.optimization-online.org/DB_HTML/2017/01/5806.html`.

[200] W. van Ackooij and R. Henrion. "(Sub-)Gradient Formulae for Probability Functions of Random Inequality Systems under Gaussian Distribution". In: *SIAM/ASA Journal on Uncertainty Quantification* 5.1 (2017), pp. 63–87.

[201] W. van Ackooij and W. de Oliveira. *Nonsmooth DC-constrained optimization: constraint qualification and minimizing methodologies*. Tech. rep. CMA - Mines ParisTech. Available at www.oliveira.mat.br, 2017.

[202] W. van Ackooij, W. de Oliveira, and Y. Song. "Adaptive Partition-Based Level Decomposition Methods for Solving Two-Stage Stochastic Programs with Fixed Recourse". In: *INFORMS Journal on Computing* 30.1 (2018), pp. 57–70.

[203] W. van Ackooij, A. Frangioni, and W. de Oliveira. "Inexact stabilized Benders' decomposition approaches with application to chance-constrained problems with finite support". In: *Computational Optimization and Applications* 65.3 (2016), pp. 637–669.

[204] A. Veinott. "The supporting hyperplane method for unimodal programming". In: *Operations Research* 15 (1967), pp. 147–152.

[205] A. Wachter and L. T. Biegler. "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming". In: *Mathematical Programming* 106.1 (2006), pp. 25–57.

[206] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. "Image quality assessment: from error visibility to structural similarity". In: *IEEE transactions on image processing* 13.4 (2004), pp. 600–612.

[207] P. Wentges. "Accelerating Benders' Decomposition for the Capacitated Facility Location Problem". In: *Mathematical Methods of Operations Research* 44.2 (1996), pp. 267–290.

[208] C. Wolf, C. I. Fábián, A. Koberstein, and L. Stuhl. "Applying oracles of on-demand accuracy in two-stage stochastic programming. A computational study". In: *European Journal of Operational Research* 239.2 (2014), pp. 437–448.

[209] C. Wolf, C. I. Fábián, A. Koberstein, and L. Stuhl. "Applying oracles of on-demand accuracy in two-stage stochastic programming. A computational study". In: *European Journal of Operational Research* 239.2 (2014), pp. 437–448.

[210] H. Xu. "Level Function Method for Quasiconvex Programming". In: *Journal of Optimization Theory and Applications* 108.2 (2001), pp. 407–437.

[211] Y. Yang and J. M. Lee. "A tighter cut generation strategy for acceleration of Benders decomposition". In: *Computers and Chemical Engineering* 44 (2012), pp. 84–93.

[212] Z. M. Zadeh and E. Khorram. "Convexity of chance constrained programming problems with respect to a new generalized concavity notion". In: *Annals of Operations Research* 196.1 (2012), pp. 651–662.

[213] G. Zakeri, A. Philpott, and D. M. Ryan. "Inexact cuts in Benders decomposition". In: *SIAM Journal on Optimization* 10.3 (2000), pp. 643–657.

[214] S. Zaourar and J. Malick. "Prices stabilization for inexact unit-commitment problems". In: *Mathematical Methods of Operations Research* 78.3 (2013), pp. 341–359.

[215] C. J. Zappe and A. V. Cabot. "The Application Of Generalized Benders Decomposition To Certain Nonconcave Programs". In: *Computers Math. Applic.* 21.6/7 (1991), pp. 181–190.

[216] S. K. Zavriev and F. V. Kostyuk. "Heavy-ball method in nonconvex optimization problems". In: *Computational Mathematics and Modeling* 4.4 (1993), pp. 336–341.