Computational Aspects of Column Generation Methods for Nonlinear Optimization

Renaud Chicoisne

Computational Optimization and Applications manuscript No.
(will be inserted by the editor)

Computational Aspects of Column Generation for Nonlinear Optimization

Renaud Chicoisne

Received: date / Accepted: date

Abstract Solving large scale nonlinear optimization problems requires either significant computing resources or the development of specialized algorithms. For Linear Programming (LP) problems, decomposition methods can take advantage of problem structure, gradually constructing the full problem by generating variables or constraints. We first present a direct adaptation of the Column Generation (CG) methodology for nonlinear optimization problems, such that when optimizing over a structured set $\mathcal{X}$ plus a moderate number of complicating constraints, we solve a succession of 1) restricted *master* problems on a smaller set $\mathcal{S} \subset \mathcal{X}$ and 2) *pricing* problems that are Lagrangean relaxations wrt the complicating constraints. The former provides feasible solutions and feeds dual information to the latter. In turn, the pricing problem identifies a variable of interest that is then taken into account into an updated subset $\mathcal{S}' \subset \mathcal{X}$.

Our approach is valid *whenever the master problem has zero Lagrangean duality gap wrt to the complicating constraints*, and not only when $\mathcal{S}$ is the convex hull of the generated variables as in CG for LPs, but also with a variety of subsets such as the conic hull, the linear span, and a special variable aggregation set. We discuss how the structure of $\mathcal{S}$ and its update mechanism influence the convergence and the difficulty of solving the restricted master problems, and present linearized schemes that alleviate the computational burden of solving the pricing problem.

We test our methods on synthetic portfolio optimization instances with up to 5 million variables including nonlinear objective functions and second order cone constraints. We show that some CGs with linearized pricing are 2-3 times faster than solving the complete problem directly and are able to provide solutions within 1% of optimality in 6 hours for the larger instances, whereas solving the complete problem runs out of memory.

Keywords Nonlinear Optimization · Column Generation · Lagrangean Duality · Portfolio Optimization

Renaud Chicoisne
ULB, Brussels, Belgium and INOCS, INRIA Lille Nord-Europe, France
OrcidId: 0000-0002-5001-4350
E-mail: renaud.chicoisne@gmail.com

1 Introduction

Decomposition methods are fundamental tools to solve difficult large scale problems. In this work, we focus on Column Generation (CG) algorithms, where the number of variables is too large to allow a direct solution with an off-the-shelf optimization software. More formally, we focus on solving the following problem:

$$(P(\mathcal{X})) \quad \omega(\mathcal{X}) := \min_{x,y} \quad f(x,y) \\ \text{s.t.} \quad x \in \mathcal{X},\ -g(x,y) \in \mathcal{C},$$

where $\mathcal{C}$ is a cone in some Euclidean space, $\mathcal{X}$ is a high-dimensional structured set and f and g are generic mappings[1]. The feasibility set of the auxiliary variables y is fully defined by constraints $-g(x,y) \in \mathcal{C}$.

Defining n, n_0 and m as the respective dimensions of x, y and $\mathcal{C}$, we assume that the main issue with $P(\mathcal{X})$ is the presence of the *difficult* or *coupling/side* constraints $-g(x,y) \in \mathcal{C}$ and the magnitude of $n \gg 1$. In other words, we consider a setting in which: 1) for some low-dimensional subset $\mathcal{S} \subseteq \mathcal{X}$, $P(\mathcal{S})$ can be solved efficiently, and 2) $P(\mathcal{X})$ without the constraints $-g(x,y) \in \mathcal{C}$ gives birth to an efficiently solvable problem. We propose to use these simpler optimization problems to build a computationally efficient solution method for $P(\mathcal{X})$. For the sake of simplicity, we do not consider equalities in the complicating constraints (the generalization is straightforward).

1.1 Preliminaries

We now introduce several definitions that are used through this paper. We call the *dual cone* of the cone $\mathcal{C}$, the set $\mathcal{C}^*$ defined as $\mathcal{C}^* := \{u : \langle u, v\rangle \geqslant 0, \forall v \in \mathcal{C}\}$. For any penalization vector $\lambda \in \mathcal{C}^*$, let us define the following *Lagrangean relaxation*:

$$(L(\mathcal{X},\lambda)) \quad \omega(\mathcal{X},\lambda) := \min_{x\in\mathcal{X},y} \{f(x,y) + \langle \lambda, g(x,y)\rangle\}.$$

For any $\lambda \in \mathcal{C}^*$ we have $\omega(\mathcal{X},\lambda) \leqslant \omega(\mathcal{X})$. The *Lagrangean dual* of $P(\mathcal{X})$ is:

$$(D(\mathcal{X})) \quad \max_{\lambda\in\mathcal{C}^*} \{\omega(\mathcal{X},\lambda)\} \leqslant \omega(\mathcal{X}).$$

We say that $P(\mathcal{X})$ has no Lagrangean *duality gap* if $P(\mathcal{X})$ and $D(\mathcal{X})$ share the same optimal value $\omega(\mathcal{X})$. Notice that the concept of Lagrangean dual is associated to the constraints that are relaxed, which in this work are the constraints $-g(x,y) \in \mathcal{C}$. On another hand, given a set $\mathcal{S} \subseteq \mathcal{X}$, we define the *restricted problem* as follows:

$$(P(\mathcal{S})) \quad \omega(\mathcal{S}) := \min_{x\in\mathcal{S},y} \{f(x,y) : -g(x,y) \in \mathcal{C}\}.$$

Given that the original problem $P(\mathcal{X})$ is a relaxation of $P(\mathcal{S})$ for any $\mathcal{S} \subseteq \mathcal{X}$, we have $\omega(\mathcal{X}) \leqslant \omega(\mathcal{S})$. If $\mathcal{S}$ contains the projection onto the x-components of an optimal solution of $P(\mathcal{X})$, notice that we have $\omega(\mathcal{X}) = \omega(\mathcal{S})$ and $P(\mathcal{S})$ returns optimal solutions for $P(\mathcal{X})$.

[1] f and g satisfy some convexity properties in Section 3.

1.2 Main concept

In this paper, we extend the concept of CG beyond the scope of LPs and simplicial approximations of $\mathcal{X}$, while keeping a similar philosophy: in the LP case, at each iteration k we solve an approximation $P(\mathcal{S}^k)$ of $P(\mathcal{X})$ that uses $\mathcal{S}^k := \text{conv}(\bar{x}^l)_{l\in\{1,\dots,k\}}$, the convex hull of a family of *columns* $\bar{x}^l$, each belonging to $\mathcal{X}$. Doing so, we obtain an upper bound $\omega(\mathcal{S}^k)$ on $\omega(\mathcal{X})$ and retrieve the corresponding optimal dual multipliers λ^k. These multipliers are fed to the *pricing problem* $L(\mathcal{X}, \lambda^k)$ that returns an optimal solution $(\bar{x}^k, \bar{y}^k)$ and provides a lower bound $\omega(\mathcal{X}, \lambda^k)$ for $\omega(\mathcal{X})$. As pictured in Figure 1, we iteratively refine both problems until the optimality gap $\omega(\mathcal{X}, \lambda^k) - \omega(\mathcal{S}^k)$ is under some tolerance. Our approach

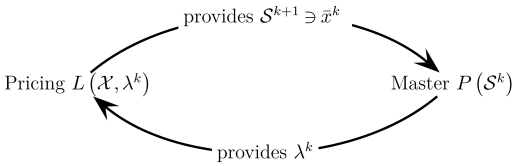

Fig. 1: CG feedback: $(\bar{x}^k, \bar{y}^k)$ is optimal for $L(\mathcal{X}, \lambda^k)$, λ^k is dual-optimal for $P(\mathcal{S}^k)$

generalizes CG in several ways: under reasonable conditions 1) the approximating set $\mathcal{S}^k$ does not have to be a convex hull of previous columns, and 2) $P(\mathcal{X})$ is not necessarily an LP but $P(\mathcal{S}^k)$ must have zero Lagrangean duality gap wrt the complicating constraints $-g(x, y) \in \mathcal{C}$. Further, 1) depending on the structure of the approximations $\mathcal{S}^k$, the master problem can be greatly simplified and 2) under some convexity assumptions, it is possible to replace $L(\mathcal{X}, \lambda^k)$ by a computationally easier pricing. We now present the assumptions used in this article.

1.3 Working hypothesis

We make two kinds of assumptions: the first ensures the validity of our framework and the remaining are necessary to make it computationally efficient:

Assumption 1 *$\mathcal{S}$ is such that $P(\mathcal{S})$ can be solved by a Lagrangean algorithm that provides a multiplier $\lambda \in \mathcal{C}^*$ such that $\omega(\mathcal{S}) = \omega(\mathcal{S}, \lambda)$.*

Assumption 2 *For any $\lambda \in \mathcal{C}^*$ we can solve efficiently $L(\mathcal{X}, \lambda)$ in practice.*

Assumption 3 *The choice of $\mathcal{S}$ makes $P(\mathcal{S})$ efficiently solvable in practice.*

Assumption 1 implies that $P(\mathcal{S})$ has no Lagrangean duality gap and we have an algorithm to find an optimal primal-dual pair $((x, y), \lambda)$ for $P(\mathcal{S})$; notice that (x, y) is also optimal for $L(\mathcal{S}, \lambda)$. Assumption 1 is satisfied by many optimization problems such as e.g. LPs or Linear Conic problems (LC) and Nonlinear optimization Problems (NLP) that satisfy a *Constraint Qualification* [38], which can be solved using an interior point method [36]. Slater's condition is a popular constraint qualification that is satisfied if the problem at hand is convex and strictly

feasible[2]: Although Assumption 1 may sound overly restrictive, we show in Proposition 1 that for a regular enough $\mathcal{S}$, $P(\mathcal{S})$ satisfies Slater's condition. Notice that in general, we only need $P(\mathcal{S})$ to satisfy Assumption 1, while $P(\mathcal{X})$ *may not*.

Assumptions 2 and 3 are not needed from a theoretical point a view, however, they are essential for our methodology to be competitive computationally. Assumption 2 is the basic assumption for classical CG for LPs and means that the pricing problem $L(\mathcal{X}, \lambda)$ is either 1) block-decomposable thanks to the structure of $\mathcal{X}$ and can be solved in parallel, or 2) there is an efficient dedicated algorithm to solve it. Finally, Assumption 3 says that $\mathcal{S}$ is e.g. low dimensional and defined with a few constraints[3].

Our objective is to design an iterative search in terms of both λ and $\mathcal{S}$ that successively improves the lower and upper bounds $\omega(\mathcal{X}, \lambda)$ and $\omega(\mathcal{S})$, returning increasingly good feasible solutions as a byproduct. Our framework achieves this goal by feeding information from one problem to the other by updating respectively λ from $P(\mathcal{S})$ and $\mathcal{S}$ from $L(\mathcal{X}, \lambda)$, while choosing computationally efficient approximations $\mathcal{S}$ and pricing problems.

1.4 Dantzig-Wolfe for LPs

To illustrate our point, consider the following LP as a special case of $P(\mathcal{X})$ - i.e. when $\mathcal{X}$ is a polyhedron, f and g are linear mappings and $\mathcal{C} := \mathbb{R}^m_+$:

$$\omega(\mathcal{X}) := \min_{x,y} \quad c^\top x + d^\top y \tag{1a}$$
$$\text{s.t.} \quad x \in \mathcal{X} := \{x \geqslant 0 : Ax = a\} \tag{1b}$$
$$Xx + Yy \geqslant b. \tag{1c}$$

As pointed out before, if we were to replace $\mathcal{X}$ by some wisely chosen subset $\mathcal{S} \subseteq \mathcal{X}$ in (1), we would obtain a cheap upper bound for the optimal value of (1).

Master problem $P(\mathcal{S})$ In this LP case, there is a natural choice for $\mathcal{S}$ readily available [7]: Letting $\mathcal{V}$ be the set of vertices of $\mathcal{X}$ and $\mathcal{R}$ a complete set of extreme rays of $\mathcal{X}$, we have $\mathcal{X} = \text{conv } \mathcal{V} + \text{cone } \mathcal{R}$, where cone $\mathcal{R}$ is the conic hull of $\mathcal{R}$:

$$\mathcal{X} = \left\{ x : x = \sum_{l:\bar{x}^l \in \mathcal{V}} \theta_l \bar{x}^l + \sum_{l:\bar{x}^l \in \mathcal{R}} \theta_l \bar{x}^l, \text{ for some } \theta \geqslant 0 : \sum_{l:\bar{x}^l \in \mathcal{V}} \theta_l = 1 \right\}.$$

Problem (1) can thus be rewritten as the following *extensive formulation*:

$$\omega(\mathcal{X}) = \min_{\theta \geqslant 0, y} \quad \sum_{l:\bar{x}^l \in \mathcal{V}} \theta_l c^\top \bar{x}^l + \sum_{l:\bar{x}^l \in \mathcal{R}} \theta_l c^\top \bar{x}^l + d^\top y \tag{2a}$$
$$\text{s.t.} \quad \sum_{l:\bar{x}^l \in \mathcal{V}} \theta_l X \bar{x}^l + \sum_{l:\bar{x}^l \in \mathcal{R}} \theta_l X \bar{x}^l + Yy \geqslant b \tag{2b}$$
$$\sum_{l:\bar{x}^l \in \mathcal{V}} \theta_l = 1 \tag{2c}$$

[2] Strict feasibility is defined in Subsection 1.7

[3] This point is explored in more detail in Sections 4 and 5.

Because $\mathcal{X} := \{x \geqslant 0 : Ax = a\}$ is convex and each $\bar{x}^l$ belongs to $\mathcal{X}$, notice that the side constraints $Xx + Yy \geqslant b$ are the only remnants of the original problem. The LP dual of the extended formulation is the following problem:

$$\omega(\mathcal{X}) = \max_{\lambda \geqslant 0, \eta} \quad b^\top \lambda + \eta \tag{3a}$$

$$\text{s.t.} \quad \left(X\bar{x}^l\right)^\top \lambda \leqslant c^\top \bar{x}^l - \eta, \qquad \forall l : \bar{x}^l \in \mathcal{V} \tag{3b}$$

$$\left(X\bar{x}^l\right)^\top \lambda \leqslant c^\top \bar{x}^l, \qquad \forall l : \bar{x}^l \in \mathcal{R} \tag{3c}$$

$$Y^\top \lambda = d. \tag{3d}$$

A direct solution of problem (2) is in general impractical as its number of variables can be exponential in (n, n_0, m). However, the *Dantzig-Wolfe* (DW) algorithm [15] offers a solution method successively generating vertices and extreme rays of the polyhedron $\mathcal{X}$. It starts with finite subsets $\overline{\mathcal{V}} \subset \mathcal{V}$ and $\overline{\mathcal{R}} \subset \mathcal{R}$ and solves a restricted master problem (2) with $\overline{\mathcal{V}}$ and $\overline{\mathcal{R}}$ instead of the full sets $\mathcal{V}$ and $\mathcal{R}$. With our notation, this restricted master problem is none other than $P(\mathcal{S})$ with $\mathcal{S} := \text{conv}\, \overline{\mathcal{V}} + \text{cone}\, \overline{\mathcal{R}}$. Making different choices for $\mathcal{S}$ and consider a broader class of optimization problems is one of the central ideas of this paper.

Pricing problem $L(\mathcal{X}, \lambda)$ Obtaining the optimal dual variables λ associated with constraints (2b) in $P(\mathcal{S})$, the Lagrangean relaxation $L(\mathcal{X}, \lambda)$ is solved:

$$\begin{aligned} &\min_{x \in \mathcal{X}, y} \left\{ c^\top x + d^\top y + \lambda^\top (b - Xx - Yy) \right\} \\ &= \min_{x \in \mathcal{X}, y} \left\{ \left(c - X^\top \lambda\right)^\top x + \left(d - Y^\top \lambda\right)^\top y \right\} + \lambda^\top b. \end{aligned}$$

By dual feasibility (3d) of λ - implying that $Y^\top \lambda = d$ - it can be rewritten

$$\lambda^\top b + \min_{x \in \mathcal{X}} \left\{ \left(c - X^\top \lambda\right)^\top x \right\}, \tag{4}$$

thus eliminating the variables y from the pricing problem. Discarding this dependency in y is, however, not always possible in a nonlinear setting. Letting $\bar{x}$ be an optimal solution of the pricing problem (4), with a slight abuse of notation we refer to an *optimal solution* to either 1) a vertex of $\mathcal{X}$ if the pricing problem in x has a bounded optimal objective value, or 2) an extreme ray of $\mathcal{X}$ otherwise. In the latter case we increment $\bar{\mathcal{R}} \leftarrow \bar{\mathcal{R}} \cup \{\bar{x}\}$ and in the former $\bar{\mathcal{V}} \leftarrow \bar{\mathcal{V}} \cup \{\bar{x}\}$, which defines the particular update mechanism used by DW. We iterate until an optimality tolerance criterion is satisfied or until we generated the complete sets $\mathcal{V}$ and $\mathcal{R}$, thus solving the full, original problem $P(\mathcal{X})$.

1.5 Decomposition methods and previous work

CG algorithms were studied in depth for LPs [15,33] or Mixed Integer Linear Programming (MILP) problems [4,49], where notoriously large MILPs could be solved by embedding DW in a branch-and-price framework [17,13]. The main

idea of DW is to exploit the structure of $\mathcal{X}$ and solve smaller problems: 1) the master problem, that works over a reduced subset $\mathcal{S} \subset \mathcal{X}$ while keeping the side constraints; and 2) a pricing problem that is still large but is computationally easy to solve thanks to the absence of side constraints.

In a nonlinear setting, several algorithms such as the Alternating Direction Method of Multipliers (ADMM) [58], the Douglas-Rachford splitting operator [19] or augmented Lagrangean algorithms [47] all make use of a special structure in $\mathcal{X}$. However, they all solve inexactly the Lagrangean dual $D(\mathcal{X})$ and do not always provide feasible solutions for $P(\mathcal{X})$. Further, proving optimality or near optimality can be tricky and a concrete stopping criterion is also not always available. Closer to a generalization of CG for NLPs, the convex simplex method [59] minimizes a nonlinear objective over a polyhedron. It can be seen as solving a master problem over a *basis* of variables and - similar to the simplex algorithm - selecting the entering variable by linearizing some penalization function. Akin to DW, the simplicial decomposition [55] solves a *linearized* master problem over a subset $\mathcal{S}$ that is the convex hull of a handful of points, and the pricing problem generating such columns is the original problem $P(\mathcal{X})$ with an objective linearized at an incumbent point. Problem-dependent CG schemes for NLPs were presented in e.g. [37] for nonlinear facility location problems that are reformulated as set partitioning problems and solved with DW, whose pricing problem is an NLP with integer variables; [14] that uses a branch-and-price scheme for sibling groups reconstruction, which is reformulated as a set covering problem for which columns are generated with quadratic optimization pricing problems. A direct extension of DW for NLPs with $\mathcal{C} = \mathbb{R}^m_+$ is introduced in [9].

In an LC setting - i.e. f and g are linear mappings but $\mathcal{C}$ is a more general cone than $\mathbb{R}^m_+$ - similar extensions of CG have been developed: [2] present a decomposition procedure for Semidefinite Programming (SDP) problems where $\mathcal{S}$ is an inner approximation of a matrix set $\mathcal{X}$, that are updated with the (matricial) "columns" generated by a separation problem tailored for SDPs. The approach has two drawbacks: 1) depending on the inner approximation chosen, the master problem can be slow to attain near optimality, and 2) the pricing problem is a handmade separation problem that uses problem-specific considerations and does not provide dual bounds. For SDPs, chordal sparsity patterns [53] are able to detect underlying substructures that can be exploited by ADMM [61,60,52], but no CG approach has been attempted so far. The presence of a special substructure being crucial for decomposition, automatic structure detection in LPs were developed in [28,6,57] so that a decomposition method can make use of it. Previous CG methods for LC have focused on gradually building the set of variables considered with problem specific algorithms that are difficult to generalize.

Other works use a different kind of set $\mathcal{S}$ for LPs. A subset $\mathcal{S}$ consisting on forcing clusters of variables to share the same value - thus aggregating the variables together - is used in [35] and [8]. This variable aggregation principle has been successfully applied to Freight routing [48], general extended formulations [49], open pit mining scheduling [8], pricing problems [3], quadratic binary knapsack problems [44], support vector machine problems [39] or in a column-and-row generation context where DW is used in combination with a constraint aggregation scheme to solve resource constrained covering problems [45]. Finally, [21,22] introduce a CG scheme for almost generic sets $\mathcal{S}$, where the new columns are generated with

steepest-descent-like pricing problems that do not take advantage of an eventual problem structure.

1.6 Article outline

Section 2 presents a CG algorithm to solve the generic NLP $P(\mathcal{X})$ with a large number of variables and nonlinear-conic side constraints. We show that it admits several existing schemes as special cases, all defined by different sets $\mathcal{S}$. We present sufficient conditions to 1) drop the optimization in the y variables for the pricing problem and 2) make sure that $P(\mathcal{S})$ has no Lagrangean duality gap. As the Lagrangean relaxation of a nonlinear optimization problem can be as hard as the problem itself, under some convexity assumptions we present in Section 3 a linearized version of the methodology making the pricing problem easier to solve. Additionally, we also prove that $L(\mathcal{X}, \lambda)$ can *always* be independent of the secondary variables y in the linearized algorithm. In Section 4, we point out the relationships of our generic schemes to existing frameworks. In Section 5 we describe the risk-averse portfolio optimization problem on which we test our algorithms, and present several computational enhancements. In Section 6, we present numerical results on large scale synthetic instances and empirically prove the usefulness of our methodology. We conclude with some remarks and the description of several ongoing works in Section 7.

1.7 Background notations

Given a set $\mathcal{U}$, we call respectively conv $\mathcal{U}$, cone $\mathcal{U}$, lin $\mathcal{U}$, aff $\mathcal{U}$, relint $\mathcal{U}$ and dim $\mathcal{U}$, the convex hull, the conic hull, the linear span, the affine span, the relative interior and the dimension of $\mathcal{U}$. The *adjoint* U^* of a linear mapping $U : \mathcal{U} \to \mathcal{V}$ is the linear operator such that $\langle Uu, v\rangle = \langle u, U^*v\rangle$ for any $(u, v) \in \mathcal{U} \times \mathcal{V}$. For given integers $p, q > 0$, we denote $[p] := \{1, ..., p\}$, $||\cdot||_q$ is the q-norm in $\mathbb{R}^p$. The *Lorentz cone* of dimension $p + 1$ is the set $\mathcal{L}_2^{p+1} := \{(u, u_0) \in \mathbb{R}^{p+1} : ||u||_2 \leqslant u_0\}$, and $\mathcal{B}(\bar{u}, \rho) := \{u \in \mathbb{R}^p : ||\bar{u} - u||_2 < \rho\}$ is the open ball of radius $\rho > 0$ centered at $\bar{u} \in \mathbb{R}^p$. A cone $\mathcal{K}$ is said to be *proper* if it is convex, closed, contains no line and relint $\mathcal{K} \neq \emptyset$. If $\mathcal{K}$ is a proper cone then $u \in$ relint $\mathcal{K}$ and $\bar{u} \in \mathcal{K}$ imply that $u + \bar{u} \in$ relint $\mathcal{K}$.

Consider some function $\varphi : \mathcal{U} \to \mathcal{V}$. For some $L > 0$ and a norm $||\cdot||$, we say that φ is L-Lipschitz if for any $(u^1, u^2) \in \mathcal{U} \times \mathcal{U}$ we have $||\varphi(u^1) - \varphi(u^2)|| \leqslant L||u^1 - u^2||$. For some cone $\mathcal{K} \subseteq \mathcal{V}$, φ is said to be $\mathcal{K}$-convex [10] if for any $t \in [0, 1]$ and any $(u^1, u^2) \in \mathcal{U} \times \mathcal{U}$, we have $t\varphi\left(u^1\right) + (1-t)\varphi\left(u^2\right) - \varphi\left(tu^1 + (1-t)u^2\right) \in \mathcal{K}$. If φ is real-valued and differentiable, we call its linear approximation at some $\bar{u} \in \mathcal{U}$ the function: $\bar{\varphi}\left[\bar{u}\right] : u \to \varphi\left(\bar{u}\right) + \left\langle \nabla\varphi\left(\bar{u}\right), u - \bar{u}\right\rangle$. Notice that $\bar{\varphi}[\bar{u}]$ is a global under estimator of φ if φ is convex. If φ is vector valued, its linear approximation is the component-wise linear approximation $\bar{\varphi}\left[\bar{u}\right](u) = \varphi\left(\bar{u}\right) + D\varphi\left(\bar{u}\right)\left(u - \bar{u}\right)$, where $D\varphi(\bar{u})$ is the Jacobian of φ at $\bar{u}$. Given a linear mapping ϕ and a mapping γ, we say that $\mathcal{U} := \{u : \phi(u) = 0, -\gamma(u) \in \mathcal{K}\}$ or $\min_{u \in \mathcal{U}} \varphi(u)$ is *strictly feasible* iff there exists u such that $\phi(u) = 0$ and $-\gamma(u) \in$ relint $\mathcal{K}$.

Unless specified otherwise, through this document $((x^k, y^k), \lambda^k)$ is an optimal primal-dual pair for $P(\mathcal{S}^k)$ and $(\bar{x}^k, \bar{y}^k)$ is an optimal solution for $L(\mathcal{X}, \lambda^k)$.

2 A Generic Column Generation Algorithm

Instead of using specific forms of feeding the pricing information to the restricted problem, we use a generic mechanism to update $\mathcal{S}$ at each iteration as described in Algorithm 1. Figure 2 summarizes the relationships between the bounds of

Algorithm 1: CG

Data: A problem $P(\mathcal{X})$
Result: An optimal solution for $P(\mathcal{X})$
1 Set $\lambda^0 = 0$, $\mathcal{S}^1 \subseteq \mathcal{X}$ contains at least one feasible solution for $P(\mathcal{X})$ and $k = 1$;
2 **repeat**
3 Solve $P(\mathcal{S}^k)$. Let (x^k, y^k) be an optimal solution;
4 Let λ^k be an optimal dual vector corresponding to the constraints $-g(x, y) \in \mathcal{C}$;
5 **if** $\lambda^k = \lambda^{k-1}$ **then**
6 **return** (x^k, y^k);
7 Solve $L(\mathcal{X}, \lambda^k)$. Let $(\bar{x}^k, \bar{y}^k)$ be an optimal solution;
8 **if** $\bar{x}^k \in \mathcal{S}^k$ **then**
9 **return** (x^k, y^k);
10 Choose a set $\mathcal{S}^{k+1} \subseteq \mathcal{X}$ containing $\bar{x}^k$;
11 $k \leftarrow k + 1$;

the problems involved in Algorithm 1. In all the "bound relationship" figures of this paper, an edge $a \to b$ means that $a \leqslant b$, the gray edges are the nontrivial relationships that apply at a stopping criterion and the gray nodes are the optimal values of the problems solved by the algorithm.

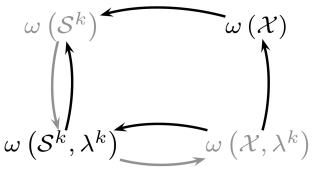

Fig. 2: Relationships of the optimal values involved in Algorithm 1.

Theorem 1 *At termination, Algorithm 1 returns an optimal solution for* $P(\mathcal{X})$.

Proof If Algorithm 1 terminates at line 8, we have $\bar{x}^k \in \mathcal{S}^k$. $(\bar{x}^k, \bar{y}^k)$ is then feasible and optimal for the Lagrangean relaxation of $P(\mathcal{S}^k)$ with λ^k:

$$\left(L(\mathcal{S}^k, \lambda^k)\right) \quad \omega\left(\mathcal{S}^k, \lambda^k\right) = \min_{x \in \mathcal{S}^k, y} \left\{ f(x, y) + \left\langle \lambda^k, g(x, y) \right\rangle \right\}.$$

In consequence, we have that $\omega(\mathcal{X}, \lambda^k) = \omega(\mathcal{S}^k, \lambda^k)$. Recall that (x^k, y^k) is optimal for $P(\mathcal{S}^k)$ and λ^k is an optimal dual vector associated to $-g(x, y) \in \mathcal{C}$ in $P(\mathcal{S}^k)$.

From Assumption 1, (x^k, y^k) is then also optimal for $L(\mathcal{S}^k, \lambda^k)$, and $\omega(\mathcal{S}^k) = \omega(\mathcal{S}^k, \lambda^k)$, thus proving the gray edges in Figure 2. To summarize, we have:

$$\omega(\mathcal{X}) \leqslant \omega\left(\mathcal{S}^k\right) = \omega\left(\mathcal{S}^k, \lambda^k\right) = \omega\left(\mathcal{X}, \lambda^k\right) \leqslant \omega(\mathcal{X}),$$

making (x^k, y^k) optimal for $P(\mathcal{X})$. Now, if Algorithm 1 terminates at line 5, we can choose $(\bar{x}^k, \bar{y}^k) = (\bar{x}^{k-1}, \bar{y}^{k-1})$ and have $\bar{x}^k \in \mathcal{S}^k$, which is the first stopping condition at line 8. □

2.1 General remarks

Algorithm 1 can be useful only if Assumptions 2 and 3 are satisfied, i.e. either $\dim \mathcal{S}^k$ is significantly smaller than n, or $P(\mathcal{S}^k)$ possesses a special structure or is sparser than in $P(\mathcal{X})$. In Sections 4 and 5, we show that the master problem can be considerably shrunk depending on the set $\mathcal{S}^k$ in use. Even though Assumption 1 must be satisfied in order to get the dual variables and make Theorem 1 hold, in presence of a nonzero duality gap Algorithm 1 can still be used as a heuristic that provides optimality bounds. Notice again that *we do not need* $P(\mathcal{X})$ to have zero duality gap, but we see in the next Subsection that it does help to make Assumption 1 hold.

We only proved that if Algorithm 1 were to stop, it would return an optimal solution, not that it would necessarily stop. Its finite termination depends on the way the sets $\mathcal{S}^k$ are generated in combination with the structure of the original problem. More precisely, the sequence $\mathcal{S}^k$ should shift towards at least one optimal solution of $P(\mathcal{X})$ by e.g. strictly growing in dimension until reaching $\mathcal{X}$ in the worst case (which can be achieved in some special cases that we describe later). Ideally, the sets $\mathcal{S}^k$ should contain increasingly good solutions for $P(\mathcal{X})$ such that we can stop prematurely the algorithm and still obtain an approximately optimal solution. This can be achieved by e.g. forcing $\mathcal{S}^{k+1}$ to contain $\mathcal{S}^k$ or x^k. Because we maintain lower and upper bounds over the optimal value of $P(\mathcal{X})$ at each iteration, early termination can be reasonably used and Algorithm 1 can provide a solution (x^k, y^k) feasible for $P(\mathcal{X})$ with an optimality gap $\omega(\mathcal{S}^k) - \omega(\mathcal{X}, \lambda^k)$.

2.2 How can the restricted problem $P(\mathcal{S})$ maintain a zero duality gap?

As we mentioned earlier, it is not clear when Assumption 1 can hold. We now show that whenever 1) the approximated sets $\mathcal{S}^k$ are described by convex constraints and are strictly feasible, and 2) $P(\mathcal{S}^k)$ admits as a feasible solution some known, feasible solution for $P(\mathcal{X})$ satisfying strictly the side constraints $-g(x, y) \in \mathcal{C}$, then $P(\mathcal{S}^k)$ satisfies Assumption 1:

Proposition 1 *Suppose that $\mathcal{C}$ is proper, g is L-Lipschitz, f is convex, we know some $(\bar{x}, \bar{y})$ such that $\bar{x} \in \mathcal{X}$ and $-g(\bar{x}, \bar{y}) \in$ relint $\mathcal{C}$, and $\mathcal{S}$ is described as*

$$\mathcal{S} := \{x : \phi(x, \theta) = 0,\ -\gamma(x, \theta) \in \mathcal{K},\ \textit{for some } \theta\},$$

for some proper cone $\mathcal{K}$, a linear mapping ϕ and a $\mathcal{K}$-convex mapping γ. If $\mathcal{S} \subseteq \mathcal{X}$ is strictly feasible and contains $\bar{x}$, then $P(\mathcal{S})$ has no Lagrangean duality gap.

Proof Appendix A.□

In this work, we consider approximated sets $\mathcal{S}^k$ that are nonempty polyhedra[4] that always contain the projection onto the variables x of *a known, strictly feasible solution for* $P(\mathcal{X})$. In fact, it is enough to find some feasible solution $(\bar{x}^0, \bar{y}^0)$ for $P(\mathcal{X})$ such that $-g(\bar{x}^0, \bar{y}^0) \in \text{relint } \mathcal{C}$, and set $\mathcal{S}^1 \ni \bar{x}^0$: In other words, we consider $\bar{x}^0$ as the first column generated.

2.3 y-independent pricing problems $L(\mathcal{X}, \lambda^k)$

Notice that $L(\mathcal{X}, \lambda^k)$ optimizes in both x *and* y and is still an NLP that can be as difficult to solve as $P(\mathcal{X})$. We partially address the former issue in the next Proposition and both in the next Section. We now introduce an adaptation of the P-property[5] [23]: Given a function $\varphi : \mathcal{U} \times \mathcal{V} \to \mathbb{R}$, consider the following NLP:

$$\min_{(u,v)\in\mathcal{U}\times\mathcal{V}} \varphi(u, v). \tag{5}$$

Problem (5) satisfies the P-property wrt v if $\min_{v\in\mathcal{V}} \varphi(u, v)$ can be solved independently of $u \in \mathcal{U}$. Even if this P-property appears to be overly restrictive, it holds if $\varphi(u, v) = \varphi_1(u, \varphi_2(v))$, for some $\varphi_2 : \mathcal{V} \to \mathbb{R}$ and some $\varphi_1 : \mathcal{U} \times \mathbb{R} \to \mathbb{R}$ that is non-decreasing in its second argument. This structure can appear if e.g. 1) φ is separable in u and v (i.e. $\varphi(u, v) = \varphi_1(u) + \varphi_2(v)$), or 2) $\varphi(u, v) = \varphi_1(u)\varphi_2(v)$ with $\varphi_1(u) \geqslant 0$ for any $u \in \mathcal{U}$. We are now ready to state a y-independency result for the Lagrangean relaxation $L(\mathcal{X}, \lambda^k)$:

Proposition 2 *If $L(\mathcal{X}, \lambda^k)$ satisfies the P-property wrt y, then $y = y^k$ is always an optimal choice in $L(\mathcal{X}, \lambda^k)$, which becomes an optimization problem in x only:*

$$\left(L\left(\mathcal{X}, \lambda^k\right)\right) \quad \omega\left(\mathcal{X}, \lambda^k\right) := \min_{x\in\mathcal{X}} \left\{ f\left(x, y^k\right) + \left\langle \lambda^k, g\left(x, y^k\right)\right\rangle \right\}.$$

Proof Appendix B. □

Proposition 2 allows to drop the optimization in y in the pricing problem if the P-property holds. We now show that using a linearized version of $L(\mathcal{X}, \lambda^k)$, we can *always* drop the optimization in y in the pricing, regardless of the P-property.

3 A linearized Column Generation Algorithm

In practice, it is common to face e.g. LPs with a nice structure that admit tailored algorithms to solve them, getting *hardened* by replacing their linear objective with a nonlinear objective function f and/or adding possibly nonlinear side constraints $-g(x, y) \in \mathcal{C}$ to their polyhedral feasible set $\mathcal{X}$. This can happen for e.g. robust optimization problems [5] that are no easier than their deterministic counterparts.

In this Section, we show that solving a pricing problem whose objective function is *linearized* at the current incumbent (x^k, y^k) holds the same guarantees as Algorithm 1, while alleviating the difficulty of solving the pricing problem whenever

[4] More precisely, projections onto the variables x of a polyhedron in an extended space.

[5] Originally used in a *Generalized Benders Decomposition* context.

optimizing a linear objective over $\mathcal{X}$ is an easy task. As we show in our experiments, this can be extremely useful whenever a linear objective pricing problem can be solved with a dedicated algorithm.

3.1 Additional assumptions and results

We now present several results that allow the use of a linearized version of $L(\mathcal{X}, \lambda^k)$ as a pricing problem. In this Section, the following extra assumption is met:

Assumption 4 *$\mathcal{S}^k$, $\mathcal{C}$ and f are convex, g is $\mathcal{C}$-convex, f and g are differentiable.*

Further, we assume that for any cost vector c, $\min_{x\in\mathcal{X}}\langle c, x\rangle$ can be solved efficiently (which is in fact equivalent to Assumption 2 if the objective function of $L(\mathcal{X}, \lambda^k)$ is linear). We now present several technical Lemmas to prove our main result:

Lemma 1 *Consider a cone $\mathcal{K}$, $\lambda \in \mathcal{K}^*$ and a $\mathcal{K}$-convex function φ. Then $\psi : u \to \langle\lambda, \varphi(u)\rangle$ is convex.*

Proof Appendix C.□

Lemma 2 *Given $\lambda \in \mathcal{V}$, a differentiable function $\varphi : \mathcal{U} \to \mathcal{V}$ and $\bar{u} \in \mathcal{U}$, the linear approximation of $\psi : u \to \langle\lambda, \varphi(u)\rangle$ at $\bar{u}$ is $\bar{\psi}[\bar{u}] : u \to \langle\lambda, \bar{\varphi}[\bar{u}](u)\rangle$.*

Proof Appendix D.□

Lemma 3 *Consider the NLC $\omega^* := \min_{u\in\mathcal{U}}\varphi(u)$, where $\mathcal{U}$ is convex and $\varphi : \mathcal{U} \to \mathbb{R}$ is differentiable. If u^* is one of its optimal solutions, it is also optimal for $\bar{\omega}[u^*] := \min_{u\in\mathcal{U}}\{\bar{\varphi}[u^*](u) := \varphi(u^*) + \langle\nabla\varphi(u^*), u - u^*\rangle\}$ and we have $\omega^* = \bar{\omega}[u^*]$.*

Proof Appendix E. □

Lemma 3 tells us that an optimal solution of a convex optimization problem is also optimal for the same problem with an objective function linearized at said solution. For any $\mathcal{S} \subseteq \mathcal{X}$, $\lambda \in \mathcal{C}^*$ and $(\bar{x}, \bar{y})$, let us define the following problem:

$$\left(\bar{L}\left[\bar{x}, \bar{y}\right](\mathcal{S}, \lambda)\right) \quad \bar{\omega}\left[\bar{x}, \bar{y}\right](\mathcal{S}, \lambda) := \min_{x\in\mathcal{S}, y}\left\{\bar{f}\left[\bar{x}, \bar{y}\right](x, y) + \langle\lambda, \bar{g}\left[\bar{x}, \bar{y}\right](x, y)\rangle\right\},$$

which is $L(\mathcal{S}, \lambda)$ with its objective function linearized at $(\bar{x}, \bar{y})$.

3.2 A linearized algorithm

Now consider the following algorithm: 1) instead of solving $L(\mathcal{X}, \lambda^k)$, the pricing we solve is its linear approximation $\bar{L}[x^k, y^k](\mathcal{X}, \lambda^k)$ at the current incumbent (x^k, y^k) of the restricted problem $P(\mathcal{S}^k)$ and 2) the stopping criterion at line 5 of Algorithm 1 is replaced by a slightly more restrictive condition. We describe the changes applied to Algorithm 1 in Algorithm 2 and illustrate in Figure 3 the relationships between the bounds of the problems involved in it.

Theorem 2 *Algorithm 2 returns an optimal solution for $P(\mathcal{X})$ at termination.*

Algorithm 2: CG-Lin changes wrt CG

5 **if** $((x^k, y^k), \lambda^k) = ((x^{k-1}, y^{k-1}), \lambda^{k-1})$ **then**
6 **return** (x^k, y^k);
7 Solve $\bar{L}[x^k, y^k](\mathcal{X}, \lambda^k)$. Let $(\bar{x}^k, \bar{y}^k)$ be an optimal solution;

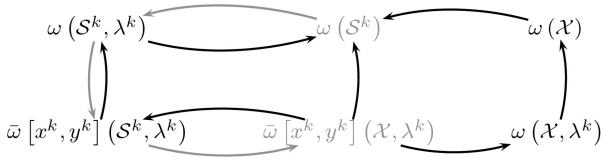

Fig. 3: Relationships of the optimal values involved in Algorithm 2.

Proof If Algorithm 2 terminates because $\bar{x}^k \in \mathcal{S}^k$, then $(\bar{x}^k, \bar{y}^k)$ is feasible and optimal for $\bar{L}[x^k, y^k](\mathcal{S}^k, \lambda^k)$ and we have:

$$\bar{\omega}\left[x^k, y^k\right]\left(\mathcal{S}^k, \lambda^k\right) = \bar{\omega}\left[x^k, y^k\right]\left(\mathcal{X}, \lambda^k\right).$$

Because f is convex and g is $\mathcal{C}$-convex, Lemmas 1 and 2 imply that $\bar{f}[x^k, y^k]$ and $\langle \lambda^k, \bar{g}[x^k, y^k](\cdot)\rangle$ are global under estimators of f and $\langle \lambda^k, g(\cdot)\rangle$ respectively. In consequence we have that

$$\bar{\omega}\left[x^k, y^k\right]\left(\mathcal{X}, \lambda^k\right) \leqslant \omega\left(\mathcal{X}, \lambda^k\right) \leqslant \omega\left(\mathcal{X}\right).$$

On another hand, (x^k, y^k) is also an optimal solution for $L(\mathcal{S}^k, \lambda^k)$ from Assumption 1. Finally, we can interpret $\overline{L}[x^k, y^k](\mathcal{S}^k, \lambda^k)$ as the linearization of $L(\mathcal{S}^k, \lambda^k)$ at one of its optimal solutions (x^k, y^k). Recalling that:

$$\begin{aligned}
\omega\left(\mathcal{S}^k, \lambda^k\right) &= \min_{x \in \mathcal{S}^k, y} \left\{ f(x,y) + \left\langle \lambda^k, g(x,y) \right\rangle \right\} \\
\bar{\omega}\left[x^k, y^k\right]\left(\mathcal{S}^k, \lambda^k\right) &= \min_{x \in \mathcal{S}^k, y} \left\{ \bar{f}\left[x^k, y^k\right](x,y) + \left\langle \lambda^k, \bar{g}\left[x^k, y^k\right](x,y) \right\rangle \right\},
\end{aligned}$$

Lemma 3 then tells us that (x^k, y^k) is also an optimal solution for $\bar{L}[x^k, y^k](\mathcal{S}^k, \lambda^k)$, giving $\omega(\mathcal{S}^k, \lambda^k) = \bar{\omega}[x^k, y^k](\mathcal{S}^k, \lambda^k)$. To summarize, we obtain

$$\begin{aligned}
\omega\left(\mathcal{X}\right) \leqslant & \omega\left(\mathcal{S}^k\right) = \omega\left(\mathcal{S}^k, \lambda^k\right) = \bar{\omega}\left[x^k, y^k\right]\left(\mathcal{S}^k, \lambda^k\right) \\
= & \bar{\omega}\left[x^k, y^k\right]\left(\mathcal{X}, \lambda^k\right) \leqslant \omega\left(\mathcal{X}, \lambda^k\right) \leqslant \omega\left(\mathcal{X}\right),
\end{aligned}$$

thus proving the optimality of (x^k, y^k) for $P(\mathcal{X})$. If Algorithm 2 stops from its criterion at line 5, we can choose $(\bar{x}^k, \bar{y}^k) = (\bar{x}^{k-1}, \bar{y}^{k-1})$, giving $\bar{x}^k \in \mathcal{S}^k$. □

3.3 y-independent pricing problems $\bar{L}[x^k, y^k](\mathcal{X}, \lambda^k)$

As opposed to Algorithm 1, we now show that regardless of the P-property, we can always get rid of the variables y in the linearized pricing $\bar{L}[x^k, y^k](\mathcal{X}, \lambda^k)$:

Proposition 3 *Taking $y = y^k$ is always optimal for $\bar{L}[x^k, y^k](\mathcal{X}, \lambda^k)$, which becomes a linear objective minimization problem in x only:*

$$\bar{\omega}\left[x^k, y^k\right]\left(\mathcal{X}, \lambda^k\right) = \min_{x \in \mathcal{X}} \left\{ \bar{f}\left[x^k, y^k\right]\left(x, y^k\right) + \left\langle \lambda^k, \bar{g}\left[x^k, y^k\right]\left(x, y^k\right)\right\rangle \right\}$$

Proof Appendix F. □

3.4 "Reduced costs"

Recall that DW for LPs can stop whenever the reduced costs[6] are zero: we now show that this is also true for the linearized scheme in our nonlinear setting.

Proposition 4 *Let $\bar{c}^k := \nabla_x f(x^k, y^k) + D^* g(x^k, y^k)\lambda^k$ be the cost vector wrt x in $\bar{L}[x^k, y^k](\mathcal{X}, \lambda^k)$. If $\bar{c}^k = 0$ then (x^k, y^k) is optimal for $P(\mathcal{X})$.*

Proof The linearized pricing in x is $\min_{x \in \mathcal{X}} \langle \bar{c}^k, x \rangle$. If $\bar{c}^k = 0$, then any $\bar{x}^k \in \mathcal{X}$ is optimal: choosing any $\bar{x}^k \in \mathcal{S}^k \subseteq \mathcal{X}$ satisfies the stopping criterion. '□

This last result provides a computationally cheap stopping criterion for the nonlinearized scheme as well, as the pricing problem - linearized or not - always provides a lower bound for $\omega(\mathcal{X})$: an all-zeroes reduced cost vector ensures that the master problem cannot be improved.

4 Relationship with existing schemes

4.1 Dantzig-Wolfe

Assume that $\mathcal{X} \subseteq \mathbb{R}^n$ is a polyhedron (that we consider bounded for simplicity), $f(x, y) := c^\top x + d^\top y$ and the conic inequality is defined by $\mathcal{C} := \mathbb{R}^m_+$ and $g(x, y) := b - Xx - Yy$. Using $\mathcal{S}^k := \text{conv}(\bar{x}^l)_{l \in \{0, \ldots, k-1\}}$ we retrieve DW for LPs. Its finite convergence is ensured by the fact that $\mathcal{X}$ has a finite - although exponential in general - number of extreme points.

Extensions Notice that if $\mathcal{C}$ is a more general cone, our algorithm generalizes DW for LCs, where at each iteration $P(\mathcal{S}^k)$ is an LC and $L(\mathcal{X}, \lambda^k)$ is an LP. Applications of DW to LCs can be found in [2] and references therein. Further, there is a direct extension of DW to a special class of nonlinear problems: going back to the general case for $\mathcal{C}$, f and g but keeping $\mathcal{X}$ polyhedral, $\bar{L}[x^k, y^k](\mathcal{X}, \lambda^k)$ turns out to be an LP. Because $\mathcal{X}$ is finitely generated, the finite convergence of the linearized algorithm is also ensured with the same arguments.

[6] In the LP case, the vector of reduced costs is $c - X^\top \lambda^k$ (see problem (4) in Subsection 1.4).

Constraint redundancy If $\mathcal{X}$ is not convex, to enforce $\mathcal{S}^k \subseteq \mathcal{X}$, we must use

$$\mathcal{S}^k := \mathcal{X} \cap \text{conv}\left(\bar{x}^l\right)_{l\in\{0,\dots,k-1\}},$$

thus potentially losing the advantage of dropping any $\mathcal{X}$-defining constraint in the master problem as in the LP/convex cases. Similarly, if the conic hull is used instead and $\mathcal{X}$ is a cone, it is sufficient to use $\mathcal{S}^k := \text{cone}(\bar{x}^l)_{l\in\{0,\dots,k-1\}}$ instead of $\mathcal{S}^k := \mathcal{X} \cap \text{cone}(\bar{x}^l)_{l\in\{0,\dots,k-1\}}$.

4.2 Bienstock-Zuckerberg (BZ)

We now link our framework with a decomposition scheme for LPs [8] where the choice of $\mathcal{S}$ differs substantially from DW. Considering a partition $\mathcal{J}^k := \{\mathcal{J}_1^k, ..., \mathcal{J}_{L_k}^k\}$ of the indices $[n]$, we force all the variables x_j belonging to a same cluster $\mathcal{J}_l^k$ to yield the same value, ultimately aggregating all the variables into a single one. In other words, we use

$$\mathcal{S}^k := \mathcal{P}(\mathcal{J}^k) := \left\{x \in \mathcal{X} : x_j = \theta_l, \forall j \in \mathcal{J}_l^k, \forall l \in [L_k], \text{ for some } \theta \in \mathbb{R}^{L_k}\right\}.$$

Update mechanism The update mechanism in this case *refines* the partition $\mathcal{J}^k$ into a new partition $\mathcal{J}^{k+1}$, by splitting some of its clusters such that the new column $\bar{x}^k$ belongs to $\mathcal{S}^{k+1} := \mathcal{P}(\mathcal{J}^{k+1})$. We call a partition *induced* by some $x \in \mathbb{R}^n$, a partition $\mathcal{J}(x) = \{\mathcal{J}_1(x), ..., \mathcal{J}_L(x)\}$ of $[n]$ such that for every $l \in [L]$ and any pair of indices $(j, j') \in \mathcal{J}_l(x) \times \mathcal{J}_l(x)$ we have $x_j = x_{j'}$. Given two partitions of $[n]$, $\mathcal{J} = \{\mathcal{J}_1, ..., \mathcal{J}_L\}$ and $\mathcal{J}' = \{\mathcal{J}'_1, ..., \mathcal{J}'_{L'}\}$, their *intersection* $\mathcal{J}\Delta\mathcal{J}'$ is the partition of $[n]$ defined as follows:

$$\mathcal{J}\Delta\mathcal{J}' := \left\{\mathcal{J}_l \cap \mathcal{J}'_{l'}, \forall \left(l, l'\right) \in [L] \times [L']\right\}.$$

Given a partition $\mathcal{J}^k$ and $\mathcal{J}^k(\bar{x}^k)$ a partition induced by $\bar{x}^k$, we first compute the refined partition $\mathcal{J}^{k+1} := \mathcal{J}^k\Delta\mathcal{J}^k(\bar{x}^k)$ and the new restricted set is given by $\mathcal{S}^{k+1} := \mathcal{P}(\mathcal{J}^{k+1})$.

Extensions and convergence Such a scheme makes Algorithms 1 and 2 generalizations to nonlinear problems of the BZ algorithm [8]. This time the convergence is not ensured by some property of $P(\mathcal{X})$, but rather thanks to the structure of the sets $\mathcal{S}^k$. In fact, either 1) the partition is refined until turning into $\{\{1\}, ..., \{n\}\}$, meaning we reached the original problem $P(\mathcal{X})$, or 2) the partition is not refined, in which case $\mathcal{J}^{k+1} = \mathcal{J}^k\Delta\mathcal{J}^k(\bar{x}^k) = \mathcal{J}^k$. It is not difficult to see that the latter implies that $\bar{x}^k \in \mathcal{S}^k = \mathcal{P}(\mathcal{J}^k)$, which is a stopping criterion for both Algorithms 1 and 2. The former implies that we are guaranteed to converge to an optimal solution in at most n iterations because the size of the partition increases by at least one (i.e. $|\mathcal{J}^{k+1}| > |\mathcal{J}^k|$). Motivating the scheme in the next Subsection, [35] show that given a sequence of columns $(\bar{x}^l)_{l\in\{0,\dots,k-1\}}$, we have $\mathcal{S}^k \supseteq \mathcal{X} \cap \text{lin}(\bar{x}^l)_{l\in\{0,\dots,k-1\}}$.

Induced partition cardinality Notice that the pricing problem may provide a column $\bar{x}^k$ with a large number of different values, hence generating a high-cardinality induced partition $\mathcal{J}(\bar{x}^k)$ and increasing rapidly the size of the partition $\mathcal{J}^{k+1}$ used in the next restricted problem $P(\mathcal{S}^{k+1})$. This issue is partially addressed by the linearized Algorithm 2 and completely circumvented in the next scheme. For example, if $\mathcal{X}$ is polyhedral and possesses the integrality property [24], the pricing problems $\bar{L}[x^k, y^k](\mathcal{X}, \lambda^k)$ can return integer optimal solutions $\bar{x}^k$, thus increasing the probability of having a reduced number of different values.

Constraint redundancy Given that $\{x : x_j = \theta_l, \forall j \in \mathcal{J}_l^k, \forall l \in [L_k], \text{ for some } \theta \in \mathbb{R}^{L_k}\}$ is not necessarily contained in $\mathcal{X}$, we need to keep the $\mathcal{X}$-defining constraints in general. This issue can sometimes be avoided for e.g. bound constraints $\ell \leqslant x \leqslant u$ that are present in the definition of $\mathcal{X}$: they become equivalent to the following $L_k \ll n$ bound constraints *in terms of* θ*:*

$$\max_{j \in \mathcal{J}_l^k} \ell_j \leqslant \theta_l \leqslant \min_{j \in \mathcal{J}_l^k} u_j \quad , \forall l \in \{1, ..., L_k\}\,. \tag{6}$$

4.3 Non-partitioned BZ

[35] link the last scheme for MILPs to another that uses at each iteration the subset of $\mathcal{X}$ spanned by $\text{lin}(\bar{x}^l)_{l \in \{0,\dots,k-1\}}$, i.e.

$$\mathcal{S}^k := \left\{ x \in \mathcal{X} : x = \sum_{l=0}^{k-1} \theta_l \bar{x}^l, \text{ for some } \theta \in \mathbb{R}^k \right\} = \mathcal{X} \cap \text{lin}(\bar{x}^l)_{l \in \{0,\dots,k-1\}}.$$

Akin to the classical BZ, it converges in at most n iterations. What is not mentioned in [35] is that this is also true independently of the structure of $P(\mathcal{X})$. Instead of using partitions of variables, it uses the raw directions $\bar{x}^l$, thus avoiding an explosive increase in the number of variables. This comes at the cost of a less structured $P(\mathcal{S})$: in fact, variable aggregation is akin to a contraction operation in combinatorial optimization, which can eliminate a substantial amount of rows and columns when dealing with structured LPs.

Notice that even if we can maintain a reasonable number of variables in the master problem, the loss in structure in comparison to BZ prohibits in general the use of the trick we present in (6), and all the $\mathcal{X}$-defining constraints must be kept, including variables bounds.

4.4 How do we check if $\bar{x}^k \in \mathcal{S}^k$?

In our experimental design, we consider the four aforementioned sets $\mathcal{S}^k$. We now show how to determine if $\bar{x}^k \in \mathcal{S}^k$ efficiently in those special cases: For BZ, it is enough to check if the size of the partition after refinement increased or not; For the linear span case, it is enough to check if $\bar{x}^k$ is a linear combination of the previous columns, which is done by projection. In the convex and conic hull cases, we must check whether a small LP is feasible. Let the polyhedron Θ^k be as follows:

$$\Theta^k := \begin{cases} \left\{\theta \in \mathbb{R}_+^k : \sum_{l=0}^{k-1} \theta_l = 1\right\} & \text{If using the convex hull} \\ \mathbb{R}_+^k & \text{If using the conic hull.} \end{cases}$$

In these cases, we have that $\bar{x}^k \in \mathcal{S}^k$ iff $\{\theta \in \Theta^k : \sum_{l=0}^{k-1} \theta_l \bar{x}^l = \bar{x}^k\} \neq \emptyset$, which can be done by solving an LP having k variables θ_k and $O(n) \gg O(1)$ linear constraints. To avoid this large number of constraints, we choose to solve the following problem instead:

$$\text{distance}_2^2\left(\bar{x}^k, \mathcal{S}^k\right) := \min_{\theta \in \Theta^k} \left|\left| \bar{x}^k - \sum_{l=0}^{k-1} \theta_l \bar{x}^l \right|\right|_2^2. \tag{7}$$

We can see that $\bar{x}^k \in \mathcal{S}^k$ iff $\text{distance}_2^2\left(\bar{x}^k, \mathcal{S}^k\right) = 0$. There are two advantages to use (7): 1) we are able to monitor the distance of the current column $\bar{x}^k$ to $\mathcal{S}^k$, and 2) when solving (7) with an interior point method, the $O(k)$ constraints that define Θ^k are not a problem because $k \ll n$ and the gradient and Hessian of the penalized objective have dimensions k and $k \times k$ respectively.

4.5 x-free master problems $P(\mathcal{S})$ and constraint redundancy

First, notice that all of the aforementioned schemes are of the form

$$\mathcal{S}^k := \left\{ x \in \mathcal{X} : x = Q^k \theta \text{ for some } \theta \in \Theta^k \right\},$$

meaning that $P(\mathcal{S}^k)$ is equivalent to the following problem in θ and y only:

$$\left(P(\mathcal{S}^k)\right) \quad \omega\left(\mathcal{S}^k\right) := \min_{\theta \in \Theta^k, y} \left\{ f(Q^k\theta, y) : Q^k\theta \in \mathcal{X}, -g\left(Q^k\theta, y\right) \in \mathcal{C} \right\}.$$

Further, we are sometimes able to shrink or make redundant some $\mathcal{X}$-defining constraints in $P(\mathcal{S}^k)$. This is of crucial importance from a computational point of view, as we must make $P(\mathcal{S}^k)$ as simple to solve as possible: Given a generic set $\mathcal{G}$, a convex set $\mathcal{V}$, a cone $\mathcal{K}$, a subspace $\mathcal{E} := \{x : Ex = 0\}$ and some variable bounds (if x is a vector) $\mathcal{B} := \{x : \ell \leqslant x \leqslant u\}$, consider that

$$\mathcal{X} := \mathcal{G} \cap \mathcal{V} \cap \mathcal{K} \cap \mathcal{E} \cap \mathcal{B}.$$

We summarize in Table 1 how each of these $\mathcal{X}$-defining constraints can be simplified depending of the scheme used:

$\mathcal{S}^k$	$\mathcal{X} \cap \text{conv}(\bar{x}^l)_l$	$\mathcal{X} \cap \text{cone}(\bar{x}^l)_l$	$\mathcal{X} \cap \text{lin}(\bar{x}^l)_l$	$\mathcal{P}(\mathcal{J}^k)$
Θ^k	$\theta \geqslant 0 : \sum_{l=0}^{k-1} \theta_l = 1$	$\mathbb{R}^k_+$	$\mathbb{R}^k$	$\mathbb{R}^{\lvert\mathcal{J}^k\rvert}$
$Q^k\theta \in \mathcal{V}$	-	as is	as is	as is
$Q^k\theta \in \mathcal{K}$	redundant if $\mathcal{K}$ is convex	-	as is	as is
$Q^k\theta \in \mathcal{E}$	-	-	-	as is but sometimes shrinkable
$Q^k\theta \in \mathcal{B}$	-	as is	as is	$\forall l \in [L_k], \begin{cases} \max_{j \in \mathcal{J}_l^k} \ell_j \leqslant \theta_l \\ \min_{j \in \mathcal{J}_l^k} u_j \geqslant \theta_l \end{cases}$

Table 1: Master constraints redundancy.

5 Risk-averse Portfolio Optimization Problem

We now describe the NLP we test our algorithms on. We consider the portfolio optimization problem of determining which assets to buy - with uncertain returns - such that 1) some risk of being rewarded a poor outcome is minimized, and 2) the variance of the return is kept under some threshold. As opposed to a classical expected value maximization model, using the variance and nonlinear risk measures makes the resulting optimization problem a large scale, nonlinear objective, SOC constrained optimization problem (See e.g. [31,16,1,30,54] for non expected value portfolio optimization).

5.1 Problem description

Consider a portfolio optimization problem where we allocate the resources of $T+1$ different clients interested in disjoint subsets of stocks and having different risk profiles. As the administrator of the budgets gets a portion of the benefits, it also requires that the variance of the overall returns must be under some threshold σ^2. T clients have budgets b_t and are interested in n_t assets each, while a separate client - the 0^{th} - has budget b_0 and is interested in $n_0 \ll n := \sum_{t=1}^{T} n_t$ assets. We purposefully separate the 0^{th} client from the others as its associated decision variables are only a handful and are modelled with our extra y variables. Client t (0) has to pay a unitary cost a_j^t (a_j^0) per asset j. Each asset j has an uncertain future value c_j^t (c_j^0) and at most u_j^t (u_j^0) units can be purchased . The variables x_j^t (y_j) represent the amount of each asset j purchased by client t (0). Defining

$$\mathcal{X}_t := \left\{ z \in \left[0, u^t\right] : \left(a^t\right)^\top z \leqslant b_t \right\}, \forall t \in \{0, ..., T\},$$

x and y must satisfy $x^t \in \mathcal{X}_t$ for each $t \in [T]$ and $y \in \mathcal{X}_0$, the latter being part of the side constraints $-g(x, y) \in \mathcal{C}$. Let us define the vectors d^t as the returns c^t, where many components are zero except for the assets that the administrator is interested in. Imposing an upper bound σ^2 on the variance of the returns of these assets is equivalent to:

$$\mathbb{V}\left(\left(d^0\right)^\top y + \sum_{t=1}^{T} \left(d^t\right)^\top x^t \right) \leqslant \sigma^2. \tag{8}$$

Each client minimizes a risk measure f_t that depends on the uncertain return of the stocks. We consider that each minimizes an entropic risk [46] of parameter α_t: $f_t(z) := \mathcal{E}_{\alpha_t}(-(c^t)^\top z)$, where[7] $\mathcal{E}_\alpha(Z) := \alpha \ln \mathbb{E}(e^{Z/\alpha})$. The general problem can be cast as follows:

$$\min_{x,y} \left\{ f_0(y) + \sum_{t=1}^{T} f_t\left(x^t\right) \,:\, x^t \in \mathcal{X}_t, \forall t \in [T],\, y \in \mathcal{X}_0,\, (8) \right\}.$$

[7] The entropic risk measure is shown to be convex in z in [46]

5.2 Sample Average Approximation (SAA)

We now approximate the last problem by using S samples (c^{ts}, c^{0s}) of respective probabilities p_s with SAA [29]. The approximations of the variance and expectations are summarized in Table 2. For simplicity, we use the same names for the functions and their respective SAAs. Defining V, V^t and V^0 such that $V^t_{sj} :=$

Original	SAA	type
$\mathcal{E}_\alpha(-c^\top z)$	$\alpha \ln \sum_{s=1}^{S} p_s e^{(-c^s)^\top z/\alpha}$	convex
$\mathbb{V}(d^\top z) \leqslant \sigma^2$	$\sum_{s=1}^{S} p_s \left((d^s)^\top z - \sum_{s'=1}^{S} p_{s'} (d^{s'})^\top z\right)^2 \leqslant \sigma^2$	quadratic (CLA) or second order cone (SOC)

Table 2: Sample Average Approximations

$\sqrt{p_s}(d^{ts}_j - d^t_j)$ and $Vx := \sum_{t=1}^{T} V^t x^t$, the variance constraint can be expressed as a classic nonlinear quadratic (CLA) convex constraint $||V^0 y + Vx||_2^2 \leqslant \sigma^2$, or the SOC constraint $(V^0 y + Vx, \sigma) \in \mathcal{L}_2^{S+1}$. The full approximated problem becomes:

$$
\begin{aligned}
\omega(\mathcal{X}) = \min_{x,y} \quad & \alpha_0 \ln \sum_{s=1}^{S} p_s e^{-\left(c^{0s}\right)^\top y/\alpha_0} + \sum_{t=1}^{T} \alpha_t \ln \sum_{s=1}^{S} p_s e^{-\left(c^{ts}\right)^\top x^t/\alpha_t} \\
\text{s.t.:} \quad & x^t \in \mathcal{X}_t \quad , \forall t \in [T] \\
& y \in \mathcal{X}_0 \\
& \begin{cases} \left|\left|V^0 y + Vx\right|\right|_2^2 \leqslant \sigma^2 & \text{If (8) is seen as a CLA} \\ \left(V^0 y + Vx, \sigma\right) \in \mathcal{L}_2^{S+1} & \text{If (8) is seen as a SOC.} \end{cases}
\end{aligned}
$$

Notice that Assumption 1 is satisfied from Proposition 1, as $P(\mathcal{X})$ is a convex NLP that for some small $\epsilon > 0$, admits the all-ϵ's vector of $\mathbb{R}^{n+n_0}$, ϵ_{n+n_0} , as a strictly feasible point and we can use ϵ_n as a starting column for $\mathcal{S}^1 \subset \mathcal{S}^2 \subset \mathcal{S}^3$....

5.3 Pricing problem

The structure of the pricing problem depends on the coupling constraint considered: Given any dual vector $(\lambda^{1,0}, \lambda^{2,0}, \lambda^{3,0}) \in \mathbb{R}^{n_0}_+ \times \mathbb{R}^{n_0}_+ \times \mathbb{R}_+$ corresponding to

the y-specific constraints $y \geqslant 0$, $y \leqslant u^0$ and $(a^0)^\top y \leqslant b_0$, the pricing problem is:

$$\begin{aligned}\omega(\mathcal{X},\lambda) = \min_{x,y} \quad & \alpha_0 \ln \sum_{s=1}^{S} p_s e^{-\left(c^{0s}\right)^\top y/\alpha_0} + \sum_{t=1}^{T} \alpha_t \ln \sum_{s=1}^{S} p_s e^{-\left(c^{ts}\right)^\top x^t/\alpha_t} \\ & - \left(\lambda^{1,0}\right)^\top y + \left(\lambda^{2,0}\right)^\top \left(y - u^0\right) + \lambda^{3,0}\left(\left(a^0\right)^\top y - b_0\right) \\ & + \begin{cases} \lambda^4\left(\left|\left|V^0 y + Vx\right|\right|_2^2 - \sigma^2\right) & \text{If (8) is seen as a CLA,} \\ & (\lambda^4 \in \mathbb{R}_+) \\ -\left(\lambda^4\right)^\top \left(V^0 y + Vx\right) - \lambda_0^4 \sigma & \text{If (8) is seen as SOC,} \\ & \left((\lambda^4, \lambda_0^4) \in \mathcal{L}_2^{S+1}\right) \end{cases} \\ \text{s.t.} \quad & x^t \in \mathcal{X}_t,\ \forall t \in [T].\end{aligned}$$

Notice that considering (8) as a SOC makes the pricing problem separable in each x^t and y and also allows the use of the y-independency result in Proposition 2. More importantly, using the linearized pricing, each problem in x^t is solvable in $O(n_t \ln n_t)$ time: the objective function becomes linear and every problem in x^t can be solved with a dedicated algorithm:

Proposition 5 *The following LP with p variables can be reduced to a continuous knapsack problem, for which an optimal solution can be found in $O(p \ln p)$ time:*

$$\omega^* := \min_{z \in [0,\mu]} \left\{\gamma^\top z : \alpha^\top z \leqslant \beta\right\}.$$

Proof Appendix G. □

We summarize In Table 3 the types of pricing problem we encounter with our framework, and how to solve them.

Linearized	Coupling cone	Type	Separable	y-independent
yes	Any	T knapsacks	yes	yes
no	$\mathbb{R}_+$	single NLP	no	no
	$\mathcal{L}_2^{S+1}$	T NLPs	yes	yes

Table 3: Types of pricing problems

5.4 Master problem

From Table 1, we summarize in Table 4 the types of sets $\mathcal{S}^k$ we use in our experiments and their implications for the master problems $P(\mathcal{S}^k)$. Notice that the convex hull and the partitioning schemes hold a clear advantage wrt the others, as their number of constraints is way lower than the rest.

$\mathcal{S}^k$	$\mathcal{X} \cap \text{conv}(\bar{x}^l)_l$	$\mathcal{X} \cap \text{cone}(\bar{x}^l)_l$	$\mathcal{X} \cap \text{lin}(\bar{x}^l)_l$	$\mathcal{P}(\mathcal{J}^k)$
$x^t \geqslant 0$	-	-	$\sum_{l=0}^{k-1} \theta_l \bar{x}^{lt} \geqslant 0$	$\theta \geqslant 0$
$x^t \leqslant u^t$	-	$\sum_{l=0}^{k-1} \theta_l \bar{x}^{lt} \leqslant u^t$	$\sum_{l=0}^{k-1} \theta_l \bar{x}^{lt} \leqslant u^t$	$\forall l \in [L_k]$, $\theta_l \leqslant \min_{(j,t)\in\mathcal{J}_l^t} u_j^t$
$(a^t)^\top x^t \leqslant b_t$	-	$\sum_{l=0}^{k-1} \theta_l (a^t)^\top \bar{x}^{lt} \leqslant b_t$	$\sum_{l=0}^{k-1} \theta_l (a^t)^\top \bar{x}^{lt} \leqslant b_t$	$\sum_{l=1}^{\lvert\mathcal{J}^k\rvert} \theta_l \sum_{(j,t)\in\mathcal{J}_l^k} a_j^t \leqslant b_t$
# var. bnds.	k	k	0	$2\lvert\mathcal{J}^k\rvert$
# lin. cnst.	1	$n+T$	$2n+T$	T

Table 4: Master constraints ($y \in \mathcal{X}_0$ and the variance constraint are always present)

Chasing the conic dual variables If we consider the variance constraint as a SOC we must be able to retrieve conic dual variables for the master problem $P(\mathcal{S})$. Even though nonlinear or linear conic solvers do exist, we could not find any general purpose package for problems having both features and returning conic multipliers. To circumvent this issue, we solve the master with an off the shelf nonlinear solver by considering the quadratic constraint as a CLA, then use the following result to obtain conic multipliers:

Proposition 6 *For some $\gamma_0 < 0$, consider the following optimization problem:*

$$\omega^* := \min_u \{\varphi(u) : \phi(u) \leqslant 0, -(\gamma(u), \gamma_0) \in \mathcal{L}_2\}, \tag{9}$$

and its equivalent representation as a classic nonlinear optimization problem:

$$\omega^* := \min_u \left\{\varphi(u) : \phi(u) \leqslant 0, ||\gamma(u)||_2^2 - \gamma_0^2 \leqslant 0\right\}. \tag{10}$$

Assume they are both convex and neither has a Lagrangean duality gap. Given an optimal primal-dual pair $(u^*, (\pi^*, \lambda^*))$ for (10) then $(u^*, (\pi^*, \hat{\lambda}, \hat{\lambda}_0))$ is an optimal primal-dual pair for (9), where: $(\hat{\lambda}, \hat{\lambda}_0) := 2\lambda^*(\gamma(u^*), -\gamma_0)$.

Proof Appendix H. □

Proposition 6 indicates that we can always derivate SOC multipliers from the multipliers of the constraint in nonlinear quadratic convex form.

5.5 Numerical Enhancement

Solving the convex optimization problems at hand *as they are* with an interior point method can rapidly exceed the capabilities of an average workstation. In fact, the Hessian matrix of the penalized objective can have many nonzero coefficients because of the entropic risk measures and the variance constraint. This observation implies that the linear system that is solved during each Newton step of the interior point method can be overly demanding both in terms of memory and running time. Introducing new variables and constraints, notice that we have the following identity: for any $t \in \{0, ..., T\}$ and any z (x^t or y) we have

$$\alpha_t \ln \sum_{s=1}^{S} p_s e^{\left(-c^{ts}\right)^\top z/\alpha_t} = \min_{v^t} \left\{\alpha_t \ln \sum_{s=1}^{S} p_s e^{v_s^t/\alpha_t} : v_s^t = -\left(c^{ts}\right)^\top z, \forall s \in [S]\right\}.$$

In the same fashion, we can replace the variance constraint with

$$||w||_2^2 \leqslant \sigma^2 \quad \text{and} \quad w = V^0 y + Vx.$$

This way, the Hessian of the Lagrangean function is a slightly larger matrix with $(T+1)S$ extra columns and rows having way less nonzero coefficients: at most $(T+1)S^2$ of them come from the entropies, and S from the variance constraint. For this reason and the fact that CG is typically useful when there are only a few side constraints, we purposefully kept S moderately small for our experiments.

Chasing the dual variables for the enhanced model We must now be able to catch the dual variables associated to the original constraints (depending only of x and y) from the dual variables of the constraints in the enhanced formulation (now also including v and w). We address this in a general setting in the next Proposition, by showing that we can ignore the extra constraints and use the multipliers *as is*:

Proposition 7 *Given a proper cone $\mathcal{K}$, consider:*

$$\omega^* := \min_u \{\varphi(u) : \phi(u) \leqslant 0, -\gamma(u) \in \mathcal{K}\}. \tag{11}$$

Suppose there is a transformation with extra variables v and w such that for any $v = Vu$, $\varphi(u) = \tilde{\varphi}(v)$ and for any $w = Wu$, $\gamma(u) = \tilde{\gamma}(w)$, so that (11) can be rewritten as:

$$\omega^* := \min_{u,v,w} \{\tilde{\varphi}(v) \,:\, \phi(u) \leqslant 0,\, -\tilde{\gamma}(w) \in \mathcal{K},\, v = Vu,\, w = Wu\}. \tag{12}$$

If both are convex and neither has a Lagrangean duality gap, given an optimal primal-dual pair $((\tilde{u}, \tilde{v}, \tilde{w}), (\tilde{\pi}, \tilde{\lambda}, \tilde{\alpha}, \tilde{\beta}))$ for (12) then $(\tilde{u}, (\tilde{\pi}, \tilde{\lambda}))$ is optimal for (11).

Proof Appendix I. □

6 Computational experience

The algorithms presented in this paper were coded in C programming language and run over Dell PowerEdge C6420 cluster nodes with Intel Xeon Gold 6152 CPUs at 2.10GHz with 32Gb RAM each. All the convex NLPs are solved using the callable library of IPOPT [43,56], using as a subroutine the linear solver Pardiso [41,42].

6.1 Methods tested and nomenclature

We test our methods 1) on different sets $\mathcal{S}$ that use the convex hull (V), the conic hull (C), the linear span (LR) or the partition-based linear span (LP), 2) using a linearized pricing problem (L) or without (NL), 3) considering the variance constraint as a conic (SOC) or as a classical nonlinear quadratic (CLA). The y-independency result or the tailored algorithm for knapsack problems are always used whenever it applies. For example, the scheme using the convex hull with linearized pricing and considering the variance constraint as a SOC will be named L-SOC-V. We summarize the different options tested in Table 5. We do not test any non-linearized scheme if the variance constraint is considered CLA, as the pricing problem is still not separable and can be as hard as $P(\mathcal{X})$.

Parameter	Possibilities
$\mathcal{S}$	V, C, LR, LP
Linearized pricing	L, NL
Variance constraint	SOC, CLA

Table 5: Algorithms tested

6.2 Instances generated

In order to push our frameworks to their limits, we generate synthetic instances of variable sizes. We consider $T \in \{1, 50\}$ blocks of equal sizes $n_t = N \in \{10^4, 10^5\}$. There are $n_0 = 50$ auxiliary variables and we generate $S = 20$ scenarios. The supplies u_j^t are uniformly drawn from $\{1, ..., 5\}$, and the costs and weights c_j^{ts} and a^t are uniformly drawn from $[1, 2]$. The budgets are $b_t = 0.05 \cdot (u^t)^\top a^t$, i.e. such that each client can buy 5% of the assets. For some scenario $s \in [S]$, letting $\hat{x}^t \in \arg\max_{x^t}\{(c^{ts})^\top x^t : x^t \in \mathcal{X}_t\}$ and $\hat{y} \in \arg\max_y\{(c^{0s})^\top y : y \in \mathcal{X}_0\}$, we set $\sigma^2 := 0.1 \cdot ||V^0\hat{y} + V\hat{x}||_2^2$ so that the variance constraint is binding.

Remark that for any value z and any utility random variable Z we have $\mathcal{E}_\alpha(-Z) = \mathcal{E}_\alpha(z - Z) - z$, meaning that minimizing the entropic risk measure means to avoid outcomes of Z such that $z - Z$ is greater than α. Given a reference random variable $\hat{Z}$, by setting $z = \mathbb{E}(\hat{Z})$ and $\alpha = \beta \cdot \sqrt{\mathbb{V}(\hat{Z})}$, the clients wish to *avoid asset selections whose outcomes can make you lose more than β standard deviations, compared to the expected return of the reference solution.* With this observation in mind, we set $\alpha_t := 0.7 \cdot ||V^t\hat{x}^t||_2$ and $\alpha_0 := 0.7 \cdot ||V^0\hat{y}||_2$. The vectors d^t are the returns c^t where all the components are zero, except for 60% of the assets bought in the referent $(\hat{y}, \hat{x})$ (i.e. only these are considered in the variance constraint), and the initial column $\bar{x}^0 \in \mathcal{S}^1$ - that is feasible for $P(\mathcal{X})$ - is:

$$\left(\bar{x}^0\right)_j^t = \begin{cases} 0 & \text{If } d_j^t \neq 0 \\ \hat{x}_j^t & \text{Otherwise.} \end{cases}$$

Absolute and relative tolerances are respectively set to 10^{-6} and 0.1% and the runs are stopped after 6 hours.

6.3 Computational results

In Tables 6, 7, 8 and 9, we report the number of iterations (it), the total execution time (t), the master time (tmas), the pricing time (tpri), the best lower bound given by a pricing problem at any time (LB), the best upper bound given by the objective value of the last master (UB), the optimality gap (gap), and the number of variables θ defined by the last set $\mathcal{S}^k$ ($|\mathcal{S}|$). The execution times are in seconds, the gaps in % and LB and UB are scaled wrt to the upper bound of L-CLA-V. The entries in bold font are the best of each column, except for the "t" column where it means that the associated scheme went faster than solving the monolithic problem. If the time limit is hit during the last iteration, we report the execution time at the end of the previous one. If an algorithm stalls before giving any partial result, we mark the entry with "*".

$\mathcal{S}$	lin	cone	it	t	tmas	tpri	LB	UB	gap	$\lvert\mathcal{S}\rvert$
V	NL	SOC	**2**	10	**0.49**	9.85	**0.9992**	**0.9992**	**0.0000**	**2**
	L	CLA	9	**5**	4.50	0.15	**0.9992**	1.0000	0.0845	9
		SOC	9	**5**	4.81	0.13	**0.9992**	1.0000	0.0845	9
C	NL	SOC	**2**	9	1.40	7.59	**0.9992**	**0.9992**	**0.0000**	**2**
	L	CLA	9	11	11.10	0.11	**0.9992**	1.0000	0.0845	9
		SOC	9	12	11.94	0.16	**0.9992**	1.0000	0.0845	9
LR	NL	SOC	**2**	11	2.92	8.34	**0.9992**	**0.9992**	**0.0000**	**2**
	L	CLA	9	18	18.27	**0.10**	**0.9992**	1.0000	0.0845	9
		SOC	9	21	20.55	0.11	**0.9992**	1.0000	0.0845	9
LP	NL	SOC	**2**	**5**	0.83	4.04	**0.9992**	**0.9992**	**0.0000**	33
	L	CLA	7	10	10.11	0.07	**0.9992**	0.9996	0.0418	208
		SOC	7	10	9.88	**0.10**	**0.9992**	0.9996	0.0418	208
Monolithic			-	8	-	-	-	**0.9992**	-	-

Table 6: Aggregated results for $T = 1$, $N = 10.000$

$\mathcal{S}$	lin	cone	it	t	tmas	tpri	LB	UB	gap	$\lvert\mathcal{S}\rvert$
V	NL	SOC	**2**	205	**3.60**	201.52	**0.9999**	**0.9999**	**0.0000**	**2**
	L	CLA	7	**29**	27.59	**0.93**	0.9998	1.0000	0.0208	7
		SOC	7	**29**	27.55	0.97	0.9998	1.0000	0.0208	7
C	NL	SOC	**2**	155	30.99	123.93	**0.9999**	**0.9999**	**0.0000**	**2**
	L	CLA	7	208	206.56	0.94	0.9998	1.0000	0.0208	7
		SOC	7	208	206.66	0.94	0.9998	1.0000	0.0208	7
LR	NL	SOC	**2**	278	105.41	172.22	**0.9999**	**0.9999**	**0.0000**	**2**
	L	CLA	7	527	525.53	0.96	0.9998	1.0000	0.0208	7
		SOC	7	527	526.32	**0.93**	0.9998	1.0000	0.0208	7
LP	NL	SOC	**2**	277	7.60	269.17	**0.9999**	**0.9999**	**0.0000**	33
	L	CLA	7	118	116.49	0.94	0.9998	1.0000	0.0158	203
		SOC	7	118	116.39	0.98	0.9998	1.0000	0.0158	203
Monolithic			-	111	-	-	-	**0.9999**	-	-

Table 7: Aggregated results for $T = 1$, $N = 100.000$

$\mathcal{S}$	lin	cone	it	t	tmas	tpri	LB	UB	gap	$\lvert\mathcal{S}\rvert$
V	NL	SOC	**2**	**481**	**22.08**	459.02	**0.9994**	**0.9994**	**0.0000**	**2**
	L	CLA	11	**339**	328.07	7.15	0.9991	1.0000	0.0858	11
		SOC	11	**345**	334.28	7.31	0.9991	1.0000	0.0858	11
C	NL	SOC	**2**	1192	719.59	471.89	**0.9994**	**0.9994**	**0.0000**	**2**
	L	CLA	11	5076	5065.19	6.66	0.9991	1.0000	0.0858	11
		SOC	11	5404	5391.71	7.12	0.9991	1.0000	0.0858	11
LR	NL	SOC	**2**	3176	2765.16	410.87	**0.9994**	**0.9994**	**0.0000**	**2**
	L	CLA	11	16221	16213.82	6.28	0.9991	1.0000	0.0858	11
		SOC	11	13514	13508.58	**5.39**	0.9991	1.0000	0.0858	11
LP	NL	SOC	**2**	2101	1561.09	538.02	**0.9994**	**0.9994**	**0.0000**	791
	L	CLA	9	6105	6094.70	5.62	0.9992	0.9996	0.0430	1763
		SOC	9	6108	6097.89	5.65	0.9992	0.9996	0.0430	1763
Monolithic			-	902	-	-	-	**0.9994**	-	-

Table 8: Aggregated results for $T = 50$, $N = 10.000$

Overall, every scheme is shown to always converge to solutions within the optimality tolerance for the small and mid-sized instances (10K-500K main variables: Tables 6, 7 and 8). Further, the execution time is mostly used to solve the master problems for the linearized schemes but more evenly split with the pricing time for the non-linearized schemes. We can see that using the convex hull with a linearized pricing (L-SOC-V and L-CLA-V) performs 2-3 times faster than the monolithic model. Along with using the partitioned linear span with a linearized pricing (L-SOC-LP and L-CLA-LP), they are the only algorithms that were able to terminate successfully or return a good quality solution in the allotted time for the largest instance (Table 9, $T = 50$, $N = 100K$) with 5 million variables. This is due to the fact that these schemes are the only ones that reduce substantially the size of our master problem, be it by making the $\mathcal{X}$ constraints redundant for the convex hull, or by shrinking the variable bounds into a small number of other variable bounds for the partitioned linear span (See Table 4 in Subsection 5.4).

$\mathcal{S}$	lin	cone	it	t	tmas	tpri	LB	UB	gap	$\lvert\mathcal{S}\rvert$
V	NL	SOC	*	*	*	*	*	*	*	*
	L	CLA	7	**2303**	**2235.15**	47.09	**0.9996**	**1.0000**	**0.0444**	**7**
		SOC	7	**2304**	2236.46	46.78	**0.9996**	**1.0000**	**0.0444**	**7**
C	NL	SOC	*	*	*	*	*	*	*	*
	L	CLA	*	*	*	*	*	*	*	*
		SOC	*	*	*	*	*	*	*	*
LR	NL	SOC	*	*	*	*	*	*	*	*
	L	CLA	*	*	*	*	*	*	*	*
		SOC	*	*	*	*	*	*	*	*
LP	NL	SOC	*	*	*	*	*	*	*	*
	L	CLA	6	**20769**	20709.51	40.40	**0.9996**	1.0008	0.1153	454
		SOC	**4**	**8115**	8079.25	**25.38**	0.9978	1.0054	0.7593	228
Monolithic			-	*	-	-	-	*	-	-

Table 9: Aggregated results for $T = 50$, $N = 100.000$

We can see that - in our case - a linearized pricing is a crucial ingredient for a successful scheme as we can use a tailored algorithm to solve it. Also, considering the main side constraint as a SOC or a CLA does not make any difference when linearizing the pricing. Even though the non-linearized schemes are not competitive for large instances, we can see that their pricing problems provide excellent quality columns and bounds and make the schemes converge in a few iterations. Notice that using the variable aggregation scheme, the master problems are significantly bigger than the others, thus making their solution slower, although the bounds they return are also significantly better.

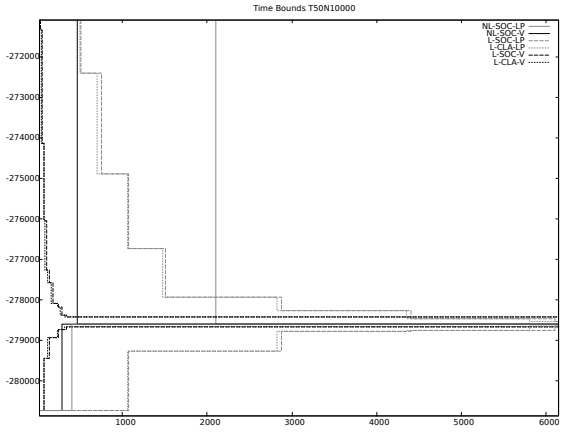

Fig. 4: Bounds Vs. Time for $T = 50$, $N = 10.000$.

In Figures 4 and 5, we show examples of progression of respectively the bounds and gaps over time of the schemes associated to V and LP on the mid-sized instance ($T = 50$, $N = 10K$). We can see that the bound/gap improvement is quite progressive for the linearized schemes, whereas it takes only a few large steps in the nonlinearized schemes.

In Figures 6 and 7, we show examples of progression of respectively the distance between $\bar{x}^k$ and $\mathcal{S}^k$, and the largest "reduced costs" (see Subsection 3.4) in absolute value over time of the schemes associated to V and LP on the mid-sized instance ($T = 50$, $N = 10K$). This empirically confirms the theoretical results about the

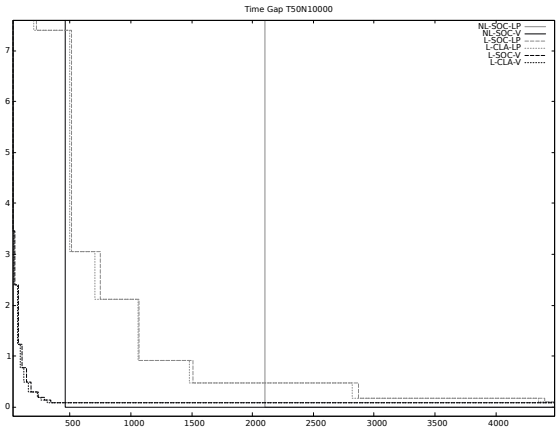

Fig. 5: Gap Vs. Time for $T = 50$, $N = 10.000$.

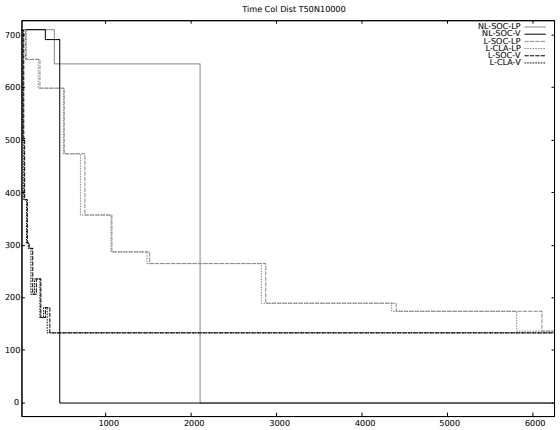

Fig. 6: Column distance to $\mathcal{S}$ Vs. Time for $T = 50$, $N = 10.000$.

stopping criterion $\bar{x}^k \in \mathcal{S}^k$ and the zero-reduced cost one presented in Proposition 4. Interestingly, to some extent these values can be used to estimate the proximity of the current solution from being optimal. We can make the same observation as for the bounds/gap, where the evolution is more progressive for the linearized schemes than the nonlinearized ones.

7 Conclusions and future work

We propose a generic primal decomposition method that unifies a broad range of existing schemes and opens the door for new exotic algorithmic frameworks. The convergence rate of the algorithms we present is not studied but can be heavily problem dependent. Several special cases of our methods have been proved to converge under mild assumptions but more work is required to prove the convergence of broader classes of algorithms. Extensive computational experiments should be

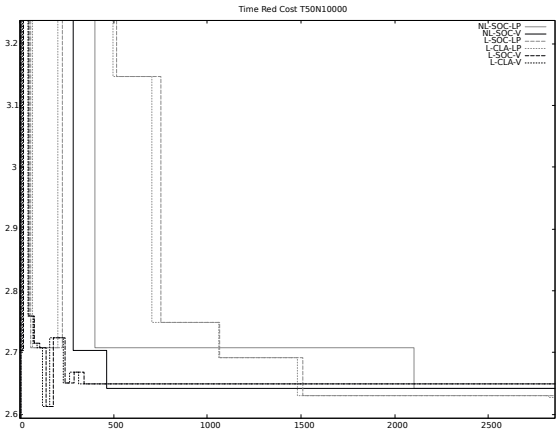

Fig. 7: Reduced Cost Vs. Time for $T = 50$, $N = 10.000$.

conducted on benchmark instances to gauge the advantages and inconvenients of each of those schemes.

Delayed column generation In this paper, we assume that some structure $\mathcal{X}$ is exploitable: in an ongoing work, we explore the same kind of algorithm presented here but *relaxing all the constraints*. It leads to algorithms sharing similarities with delayed column generation [7] and the simplex algorithm for nonlinear problems [59]. The pricing problem boils down to check if the reduced costs are zero and pick as an entering column a variable whose reduced cost is nonzero, whereas the master problem can be significantly harder than in our current setting.

Non Lagrangean relaxations as pricing problems Instead of relying on Lagrangean duality, be it in the information we have access to when solving the master problem or the kind of pricing problem we solve, different relaxations - hence dualities - can be used to provide stronger bounds and also attenuate unstable behaviors [33, 40]. We can use for example surrogate relaxations [27, 25] where instead of relaxing the side constraints in the objective, we bundle them into a single aggregated constraint $\langle \lambda, g(x, y) \rangle \leqslant 0$. The pricing problem becomes harder in general, but provides stronger dual bounds with weaker working hypothesis. We thus must be able to solve the master problems with a *surrogate algorithm* that returns optimal surrogate multipliers λ (See e.g. [34]).

Non-convex problems $P(\mathcal{X})$ The set $\mathcal{X}$ can be a tractable relaxation of a combinatorial problem $P(\widetilde{\mathcal{X}})$: the choice of $\mathcal{S}$ defines the relaxation $\mathcal{X}$ of $\widetilde{\mathcal{X}}$ we are working on. The strength of the bounds and the difficulty to solve the problems used in our algorithms can vary greatly from one $\mathcal{S}$ to another. Many NP-hard problems can be cast as *completely positive programs* [11], that admit several tractable relaxations on different matrix sets such as the cone of *doubly non-negative* matrices [18, 51]. Further, our schemes allow to solve stronger but harder SDP relaxations of combinatorial problems [32, 26].

Dual decomposition In an ongoing work, we provide *Dual decomposition schemes* where - using the tight relationship between DW and the Benders decomposition method - we present a generic constraint generation methodology where the dual variables are decomposed and generated on the fly. As in this paper, a variety of sets $\mathcal{S}$ can be used, yielding different master problems e.g. the generalized Benders decomposition [23] or constraint aggregation schemes ([20,50] or [12]).

Acknowledgements The author thanks Victor Bucarey, Bernard Fortz, Bernard Gendron, Gonzalo Muñoz, Fernando Ordóñez and Dana Pizarro for their valuable comments on an early version of this work. Powered@NLHPC: This research was partially supported by the supercomputing infrastructure of the NLHPC (ECM-02).

Conflict of interest

The author declares that he has no conflict of interest.

Appendix A Proof of Proposition 1

We prove that some $(\tilde{x}, \tilde{y})$ is strictly feasible for $P(\mathcal{S})$. Because $-g(\bar{x}, \bar{y}) \in$ relint $\mathcal{C}$ and $\mathcal{C}$ is proper, there is $\rho > 0$ such that $\mathcal{B}(-g(\bar{x}, \bar{y}), \rho) \subset$ relint $\mathcal{C}$. Consider $(\hat{x}, \hat{\theta})$ strictly feasible for $\mathcal{S}$ and $\epsilon > 0$ and $\tilde{x}$ as follows:

$$\epsilon = \min\left\{1, \frac{\rho}{L||\hat{x} - \bar{x}||}\right\}, \quad \tilde{x} = \bar{x} + \epsilon(\hat{x} - \bar{x}).$$

Because $\bar{x} \in \mathcal{S}$ and $(\hat{x}, \hat{\theta})$ is strictly feasible for $\mathcal{S}$, there exists $\bar{\theta}$ such that $\phi(\bar{x}, \bar{\theta}) = 0$, $-\gamma(\bar{x}, \bar{\theta}) \in \mathcal{K}$, $\phi(\hat{x}, \hat{\theta}) = 0$ and $-\gamma(\hat{x}, \hat{\theta}) \in$ relint $\mathcal{K}$. ϕ being linear, defining $\tilde{\theta} = \bar{\theta} + \epsilon(\hat{\theta} - \bar{\theta})$, we immediately have $\phi(\tilde{x}, \tilde{\theta}) = 0$. Further, $\epsilon > 0$, $\mathcal{K}$ is proper, $-\gamma(\bar{x}, \bar{\theta}) \in \mathcal{K}$ and $-\gamma(\hat{x}, \hat{\theta}) \in$ relint $\mathcal{K}$ so we obtain $-\epsilon\gamma(\hat{x}, \hat{\theta}) - (1 - \epsilon)\gamma(\bar{x}, \bar{\theta}) \in$ relint $\mathcal{K}$. Because γ is $\mathcal{K}$-convex and $\epsilon \in]0, 1]$ we obtain $-\gamma(\tilde{x}, \tilde{\theta}) \in$ relint $\mathcal{K}$. We just proved that $(\tilde{x}, \tilde{\theta})$ is strictly feasible for $\mathcal{S}$. We finish by proving that $(\tilde{x}, \bar{y})$ is strictly feasible for $P(\mathcal{S})$: By definition we have $\rho \geqslant L\epsilon||\hat{x} - \bar{x}|| = L||\tilde{x} - \bar{x}|| = L||(\tilde{x} - \bar{x}; \bar{y} - \bar{y})||$. Because g is L-Lipschitz, we obtain that $\rho \geqslant ||g(\tilde{x}, \bar{y}) - g(\bar{x}, \bar{y})||$, meaning that $-g(\tilde{x}, \bar{y}) \in \mathcal{B}(-g(\bar{x}, \bar{y}), \rho) \subset$ relint $\mathcal{C}$. Because $P(\mathcal{S})$ is convex, Slater's condition holds, thus finishing the proof □

Appendix B Proof of Proposition 2

If $L(\mathcal{X}, \lambda^k)$ satisfies the P-property wrt y, the problem in y given some $x \in \mathcal{X}$ is equivalent to the problem in y given $x = x^k$, i.e. for any $x \in \mathcal{X}$ we have

$$\begin{aligned} &\arg\min_y \left\{ f(x, y) + \left\langle \lambda^k, g(x, y) \right\rangle \right\} \\ &= \arg\min_y \left\{ f\left(x^k, y\right) + \left\langle \lambda^k, g\left(x^k, y\right) \right\rangle \right\}. \end{aligned} \tag{13}$$

Because $P(\mathcal{S}^k)$ satisfies Assumption 1, (x^k, y^k) is optimal for $L(\mathcal{S}^k, \lambda^k)$, i.e.

$$\omega(\mathcal{S}^k, \lambda^k) = \min_{x \in \mathcal{S}^k, y} \left\{ f(x, y) + \left\langle \lambda^k, g(x, y) \right\rangle \right\} = f\left(x^k, y^k\right) + \left\langle \lambda^k, g\left(x^k, y^k\right) \right\rangle.$$

In consequence, $y^k \in \arg\min_y\{f(x^k,y)+\langle\lambda^k, g(x^k,y)\rangle\}$. Using equality (13) finishes the proof. □

Appendix C Proof of Lemma 1

First, for any $t\in[0,1]$ and any pair $(u^1,u^2)\in\mathcal{U}\times\mathcal{U}$ we have: $t\varphi(u^1)+(1-t)\varphi(u^2)-\varphi(tu^1+(1-t)u^2)\in\mathcal{K}$. Given that $\lambda\in\mathcal{K}^*$ we have $\langle\lambda, t\varphi(u^1)+(1-t)\varphi(u^2)-\varphi(tu^1+(1-t)u^2)\rangle\geqslant 0$, meaning that

$$t\underbrace{\left\langle\lambda,\varphi\left(u^1\right)\right\rangle}_{\psi(u^1)}+(1-t)\underbrace{\left\langle\lambda,\varphi\left(u^2\right)\right\rangle}_{\psi(u^2)}\geqslant\underbrace{\left\langle\lambda,\varphi\left(tu^1+(1-t)u^2\right)\right\rangle}_{\psi(tu^1+(1-t)u^2)} \quad .\square$$

Appendix D Proof of Lemma 2

The linear approximation of ψ at $\bar{u}$ is $\bar{\psi}[\bar{u}](u):=\psi(\bar{u})+\langle\nabla\psi[\bar{u}],u-\bar{u}\rangle$. By definition, for every $u\in\mathcal{U}$:

$$\begin{aligned}\langle\nabla\psi\left[\bar{u}\right],u-\bar{u}\rangle&=\lim_{\epsilon\to0}\frac{\psi\left(\bar{u}+\epsilon\left(u-\bar{u}\right)\right)-\psi\left(\bar{u}\right)}{\epsilon}\\&=\lim_{\epsilon\to0}\frac{\langle\lambda,\varphi\left(\bar{u}+\epsilon\left(u-\bar{u}\right)\right)\rangle-\langle\lambda,\varphi\left(\bar{u}\right)\rangle}{\epsilon}\\&=\left\langle\lambda,\lim_{\epsilon\to0}\frac{\varphi\left(\bar{u}+\epsilon\left(u-\bar{u}\right)\right)-\varphi\left(\bar{u}\right)}{\epsilon}\right\rangle=\langle\lambda,D\varphi\left(\bar{u}\right)\left(u-\bar{u}\right)\rangle,\end{aligned}$$

implying that $\bar{\psi}[\bar{u}](u)=\langle\lambda,\varphi(\bar{u})\rangle+\langle\lambda,D\varphi(\bar{u})(u-\bar{u})\rangle=\langle\lambda,\bar{\varphi}[\bar{u}](u)\rangle$. □

Appendix E Proof of Lemma 3

By convexity of $\mathcal{U}$ and optimality of u^*, for any $\epsilon\in]0,1]$ and any $u\in\mathcal{U}$ we have $\varphi(u^*)\leqslant\varphi(u^*+\epsilon(u-u^*))$, i.e.

$$\frac{\varphi\left(u^*+\epsilon\left(u-u^*\right)\right)-\varphi\left(u^*\right)}{\epsilon}\geqslant 0,\quad\forall\epsilon\in]0,1],\forall u\in\mathcal{U},$$

which implies that when $\epsilon\to0$, for any $u\in\mathcal{U}$ we have $\langle\nabla\varphi\left(u^*\right),u-u^*\rangle\geqslant0$, i.e.

$$\underbrace{\varphi\left(u^*\right)+\left\langle\nabla\varphi\left(u^*\right),u-u^*\right\rangle}_{\bar{\varphi}[u^*](u)}\geqslant\underbrace{\varphi\left(u^*\right)+\left\langle\nabla\varphi\left(u^*\right),u^*-u^*\right\rangle}_{\bar{\varphi}[u^*](u^*)},\quad\forall u\in\mathcal{U},$$

meaning that u^* is optimal for $\min_{u\in\mathcal{U}}\bar{\varphi}[u^*](u)$ and implying $\omega^*=\bar{\omega}[u^*]$. □

Appendix F Proof of Proposition 3

First, notice that the following holds:

$$\begin{aligned}\bar{f}\left[x^k,y^k\right](x,y) =& f\left(x^k,y^k\right)\\ &+\left\langle\nabla_x f\left(x^k,y^k\right),x-x^k\right\rangle+\left\langle\nabla_y f\left(x^k,y^k\right),y-y^k\right\rangle\\ \bar{g}\left[x^k,y^k\right](x,y) =& g\left(x^k,y^k\right)+D_x g\left(x^k,y^k\right)\left(x-x^k\right)+D_y g\left(x^k,y^k\right)\left(y-y^k\right)\end{aligned}$$

In consequence, we have that:

$$\begin{aligned}\bar{\omega}\left[x^k,y^k\right]\left(\mathcal{X},\lambda^k\right) =& f\left(x^k,y^k\right)-\left\langle\nabla_x f\left(x^k,y^k\right),x^k\right\rangle-\left\langle\nabla_y f\left(x^k,y^k\right),y^k\right\rangle\\ &+\left\langle\lambda^k,g\left(x^k,y^k\right)-D_x g\left(x^k,y^k\right)x^k-D_y g\left(x^k,y^k\right)y^k\right\rangle\\ &+\min_{x\in\mathcal{X}}\left\{\left\langle\nabla_x f\left(x^k,y^k\right),x\right\rangle+\left\langle\lambda^k,D_x g\left(x^k,y^k\right)x\right\rangle\right\}\\ &+\min_{y}\left\{\left\langle\nabla_y f\left(x^k,y^k\right),y\right\rangle+\left\langle\lambda^k,D_y g\left(x^k,y^k\right)y\right\rangle\right\}, \quad (14)\end{aligned}$$

which is separable in x and y. Wlog we can assume that (14) attains its minimum, otherwise the master problem is unbounded and we can stop. Under this assumption, the first order optimality conditions in y for the master problem $P(\mathcal{S}^k)$ are $\nabla_y f(x^k,y^k)+D_y g(x^k,y^k)^*\lambda^k=0$, in turn implying that for any y: $\langle\nabla_y f(x^k,y^k),y\rangle+\langle\lambda^k,D_y g(x^k,y^k)y\rangle=0$, meaning that the objective function of (14) is identically zero. □

Appendix G Proof of Proposition 5

Let us define the following subsets of $[p]$:

$$\begin{aligned}\mathcal{J}_+ &:= \{j\in[p]:\gamma_j<0,\alpha_j>0\}, \quad \mathcal{J}_- := \{j\in[p]:\gamma_j>0,\alpha_j<0\},\\ \mathcal{J}_0 &:= \{j\in[p]:\gamma_j,\alpha_j\geqslant 0\}, \quad \mathcal{J}_\mu := \{j\in[p]:\gamma_j,\alpha_j\leqslant 0\}.\end{aligned}$$

First notice that we can fix beforehand the following variables:

$$z_j^* = \begin{cases}\mu_j & \text{If } j\in\mathcal{J}_\mu\\ 0 & \text{If } j\in\mathcal{J}_0.\end{cases}$$

Next, for every $j\in\mathcal{J}_-$, we use the change of variable $z_j\leftarrow\mu_j-z_j$, obtaining the following problem:

$$\begin{aligned}\omega^* := \sum_{j\in\mathcal{J}_-\cup\mathcal{J}_\mu}\gamma_j\mu_j+\min_z\quad & \sum_{j\in\mathcal{J}_+\cup\mathcal{J}_-}\hat{\gamma}_j z_j\\ \text{s.t.:}\quad & \sum_{j\in\mathcal{J}_+\cup\mathcal{J}_-}\hat{\alpha}_j z_j\leqslant\hat{\beta}\\ & z_j\in[0,\mu_j], \qquad \forall j\in\mathcal{J}_+\cup\mathcal{J}_-\end{aligned}$$

where $\hat{\beta} := \beta - \sum_{j \in \mathcal{J}_- \cup \mathcal{J}_\mu} \alpha_j \mu_j$ and:

$$\hat{\alpha}_j := \begin{cases} \alpha_j & \text{if } j \in \mathcal{J}_+ \\ -\alpha_j & \text{if } j \in \mathcal{J}_- \end{cases} \qquad \hat{\gamma}_j := \begin{cases} \gamma_j & \text{if } j \in \mathcal{J}_+ \\ -\gamma_j & \text{if } j \in \mathcal{J}_-. \end{cases}$$

The latter is a knapsack problem with positive capacity $\hat{\beta}$ and weights $\hat{\alpha}_j$ that can be solved by sorting the remaining indices $j \in \mathcal{J}_+ \cup \mathcal{J}_-$ in increasing *disutility* $\hat{\gamma}_j / \hat{\alpha}_j$ and filling the capacity constraint until no variable is available or the capacity constraint is tight. □

Appendix H Proof of Proposition 6

It is enough to show that $(u^*, (\pi^*, \hat{\lambda}, \hat{\lambda}_0))$ satisfies the KKT conditions for (9):

$$\phi\left(u^*\right) \leqslant 0, \quad -\left(\gamma\left(u^*\right), \gamma_0\right) \in \mathcal{L}_2 \tag{15a}$$

$$\pi^* \geqslant 0, \quad \phi\left(u^*\right)^\top \pi^* = 0 \tag{15b}$$

$$\left(\hat{\lambda}, \hat{\lambda}_0\right) \in \mathcal{L}_2 \tag{15c}$$

$$\gamma\left(u^*\right)^\top \hat{\lambda} + \gamma_0 \hat{\lambda}_0 = 0 \tag{15d}$$

$$\nabla\varphi\left(u^*\right) + D\phi\left(u^*\right)^\top \pi^* + D\gamma\left(u^*\right)^\top \hat{\lambda} = 0. \tag{15e}$$

(15a) and (15b) are trivially satisfied. Given that $-\gamma_0, \lambda^* \geqslant 0$, the remaining conditions are equivalent to:

$$\left(\lambda^*\right)^2 \left(\left|\left|\gamma\left(u^*\right)\right|\right|_2^2 - \gamma_0^2\right) \leqslant 0$$

$$\lambda^* \left(\left|\left|\gamma\left(u^*\right)\right|\right|_2^2 - \gamma_0^2\right) = 0$$

$$\nabla\varphi\left(u^*\right) + D\phi\left(u^*\right)^\top \pi^* + 2\lambda^* D\gamma\left(u^*\right)^\top \gamma\left(u^*\right) = 0,$$

which are all implied by the KKT conditions for (10) at $(u^*, (\pi^*, \lambda^*))$. □

Appendix I Proof of Proposition 7

The KKT conditions for (12) are

$$\phi\left(\tilde{u}\right) \leqslant 0, \quad -\tilde{\gamma}\left(\tilde{w}\right) \in \mathcal{K}, \quad \tilde{v} = V\tilde{u}, \quad \tilde{w} = W\tilde{u} \tag{16a}$$

$$\tilde{\pi} \geqslant 0, \quad \tilde{\lambda} \in \mathcal{K}^* \tag{16b}$$

$$\phi\left(\tilde{u}\right)^\top \tilde{\pi} = 0, \quad \left\langle \tilde{\gamma}\left(\tilde{w}\right), \tilde{\lambda} \right\rangle = 0 \tag{16c}$$

$$\nabla\tilde{\varphi}\left(\tilde{v}\right) - \tilde{\alpha} = 0, \quad D\tilde{\gamma}\left(\tilde{w}\right)^* \tilde{\lambda} - \tilde{\beta} = 0 \tag{16d}$$

$$D\phi\left(\tilde{u}\right)^\top \tilde{\pi} + V^*\tilde{\alpha} + W^*\tilde{\beta} = 0. \tag{16e}$$

Conditions (16a)-(16b)-(16c) represent respectively primal and dual feasibility and complementary slackness for (11) at $(\tilde{u}, (\tilde{\pi}, \tilde{\lambda}))$. We now prove that (16d)-(16e)

imply the last remaining KKT condition for (11): stationarity. Replacing (16d) in (16e) we obtain:

$$D\phi\left(\tilde{u}\right)^{\top}\tilde{\pi}+V^{*}\nabla\tilde{\varphi}\left(\tilde{v}\right)+W^{*}D\tilde{\gamma}\left(\tilde{w}\right)^{*}\tilde{\lambda}=0. \tag{17}$$

For any (u,v,w) such that $Vu=v$ and $Wu=w$, we have $\varphi(u)=\tilde{\varphi}(Vu)$ and $\gamma(u)=\tilde{\gamma}(Wu)$, implying that

$$\nabla\varphi(u)=V^{*}\nabla\tilde{\varphi}\left(Vu\right)=V^{*}\nabla\tilde{\varphi}\left(v\right) \tag{18a}$$

$$D\gamma(u)=D\tilde{\gamma}\left(Wu\right)W=D\tilde{\gamma}\left(w\right)W. \tag{18b}$$

Using (18) at $(u,v,w)=(\tilde{u},\tilde{v},\tilde{w})$ and replacing in (17) we get $D\phi(\tilde{u})^{\top}\tilde{\pi}+\nabla\varphi(\tilde{u})+D\gamma(\tilde{u})^{*}\tilde{\lambda}=0$. □

References

1. Acerbi, C., Simonetti, P.: Portfolio optimization with spectral measures of risk. arXiv preprint cond-mat/0203607 (2002)
2. Ahmadi, A.A., Dash, S., Hall, G.: Optimization over structured subsets of positive semidefinite matrices via column generation. Discrete Optimization **24**, 129–151 (2017)
3. Álvarez, C., Mancilla-David, F., Escalona, P., Angulo, A.: A bienstock–zuckerberg-based algorithm for solving a network-flow formulation of the convex hull pricing problem. IEEE Transactions on Power Systems **35**(3), 2108–2119 (2019)
4. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., Vance, P.H.: Branch-and-price: Column generation for solving huge integer programs. Operations research **46**(3), 316–329 (1998)
5. Ben-Tal, A., El Ghaoui, L., Nemirovski, A.: Robust optimization. Princeton University Press (2009)
6. Bergner, M., Caprara, A., Ceselli, A., Furini, F., Lübbecke, M.E., Malaguti, E., Traversi, E.: Automatic dantzig–wolfe reformulation of mixed integer programs. Mathematical Programming **149**(1-2), 391–424 (2015)
7. Bertsimas, D., Tsitsiklis, J.N.: Introduction to linear optimization, vol. 6. Athena Scientific Belmont, MA (1997)
8. Bienstock, D., Zuckerberg, M.: A new LP algorithm for precedence constrained production scheduling. Optimization Online pp. 1–33 (2009)
9. Bonnans, J.F., Gilbert, J.C., Lemaréchal, C., Sagastizábal, C.A.: Numerical optimization: theoretical and practical aspects. Springer Science & Business Media (2006)
10. Boyd, S., Vandenberghe, L.: Convex optimization. Cambridge university press (2004)
11. Burer, S.: On the copositive representation of binary and continuous nonconvex quadratic programs. Mathematical Programming **120**(2), 479–495 (2009)
12. Chicoisne, R., Ordoñez, F., Espinoza, D.: Risk averse shortest paths: A computational study. INFORMS Journal on Computing **30**(3), 539–553 (2018)
13. Choi, E., Tcha, D.W.: A column generation approach to the heterogeneous fleet vehicle routing problem. Computers & Operations Research **34**(7), 2080–2095 (2007)
14. Chou, C.A., Liang, Z., Chaovalitwongse, W.A., Berger-Wolf, T.Y., DasGupta, B., Sheikh, S., Ashley, M.V., Caballero, I.C.: Column-generation framework of nonlinear similarity model for reconstructing sibling groups. INFORMS Journal on Computing **27**(1), 35–47 (2015)
15. Dantzig, G.B., Wolfe, P.: The decomposition algorithm for linear programs. Econometrica: Journal of the Econometric Society pp. 767–778 (1961)
16. Dentcheva, D., Ruszczyński, A.: Portfolio optimization with stochastic dominance constraints. Journal of Banking & Finance **30**(2), 433–451 (2006)
17. Desaulniers, G., Desrosiers, J., Dumas, Y., Marc, S., Rioux, B., Solomon, M.M., Soumis, F.: Crew pairing at air france. European journal of operational research **97**(2), 245–259 (1997)
18. Dong, H., Anstreicher, K.: Separating doubly nonnegative and completely positive matrices. Mathematical Programming **137**(1-2), 131–153 (2013)

19. Eckstein, J., Bertsekas, D.P.: On the douglas-rachford splitting method and the proximal point algorithm for maximal monotone operators. Mathematical Programming **55**(1-3), 293–318 (1992)
20. Espinoza, D., Moreno, E.: A primal-dual aggregation algorithm for minimizing conditional value-at-risk in linear programs. Computational Optimization and Applications **59**(3), 617–638 (2014)
21. García, R., Marín, A., Patriksson, M.: Column generation algorithms for nonlinear optimization, I: Convergence analysis. Optimization **52**(2), 171–200 (2003)
22. García, R., Marín, A., Patriksson, M.: Column generation algorithms for nonlinear optimization, II: Numerical investigations. Computers & Operations Research **38**(3), 591–604 (2011)
23. Geoffrion, A.: Generalized benders decomposition. Journal of optimization theory and applications **10**(4), 237–260 (1972)
24. Giles, F.R., Pulleyblank, W.R.: Total dual integrality and integer polyhedra. Linear algebra and its applications **25**, 191–196 (1979)
25. Glover, F.: Surrogate constraint duality in mathematical programming. Operations Research **23**(3), 434–451 (1975)
26. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. Journal of the ACM (JACM) **42**(6), 1115–1145 (1995)
27. Greenberg, H., Pierskalla, W.: Surrogate mathematical programming. Operations Research **18**(5), 924–939 (1970)
28. Khaniyev, T., Elhedhli, S., Erenay, F.S.: Structure detection in mixed-integer programs. INFORMS Journal on Computing **30**(3), 570–587 (2018)
29. Kleywegt, A.J., Shapiro, A., Homem-de Mello, T.: The sample average approximation method for stochastic discrete optimization. SIAM Journal on Optimization **12**(2), 479–502 (2002)
30. Krokhmal, P., Palmquist, J., Uryasev, S.: Portfolio optimization with conditional value-at-risk objective and constraints. Journal of risk **4**, 43–68 (2002)
31. Levy, H., Markowitz, H.M.: Approximating expected utility by a function of mean and variance. The American Economic Review **69**(3), 308–317 (1979)
32. Lovász, L.: On the Shannon capacity of a graph. IEEE Transactions on Information theory **25**(1), 1–7 (1979)
33. Lübbecke, M., Desrosiers, J.: Selected topics in column generation. Operations research **53**(6), 1007–1023 (2005)
34. Müller, B., Muñoz, G., Gasse, M., Gleixner, A., Lodi, A., Serrano, F.: On generalized surrogate duality in mixed-integer nonlinear programming. In: International Conference on Integer Programming and Combinatorial Optimization, pp. 322–337. Springer (2020)
35. Muñoz, G., Espinoza, D., Goycoolea, M., Moreno, E., Queyranne, M., Rivera, O.: A study of the Bienstock–Zuckerberg algorithm: applications in mining and resource constrained project scheduling. Computational Optimization and Applications **69**(2), 501–534 (2018)
36. Nesterov, Y., Nemirovskii, A.: Interior-point polynomial algorithms in convex programming. SIAM (1994)
37. Ni, W., Shu, J., Song, M., Xu, D., Zhang, K.: A branch-and-price algorithm for facility location with general facility cost functions. INFORMS Journal on Computing **33**(1), 86–104 (2021)
38. Nocedal, J., Wright, S.: Numerical optimization. Springer Science & Business Media (2006)
39. Park, Y.W.: Optimization for l 1-norm error fitting via data aggregation. Informs Journal on Computing **33**(1), 120–142 (2021)
40. Pessoa, A., Sadykov, R., Uchoa, E., Vanderbeck, F.: Automation and combination of linear-programming based stabilization techniques in column generation. INFORMS Journal on Computing **30**(2), 339–360 (2018)
41. Petra, C.G., Schenk, O., Anitescu, M.: Real-time stochastic optimization of complex energy systems on high-performance computers. Computing in Science & Engineering **16**(5), 32–42 (2014)
42. Petra, C.G., Schenk, O., Lubin, M., Gärtner, K.: An augmented incomplete factorization approach for computing the schur complement in stochastic optimization. SIAM Journal on Scientific Computing **36**(2), C139–C162 (2014)
43. Pirnay, H., Lopez-Negrete, R., Biegler, L.: Optimal sensitivity based on ipopt. Mathematical Programming Computations **4**(4), 307–331 (2012)

44. Pisinger, W.D., Rasmussen, A.B., Sandvik, R.: Solution of large quadratic knapsack problems through aggressive reduction. INFORMS Journal on Computing **19**(2), 280–290 (2007)
45. Porumbel, D., Clautiaux, F.: Constraint aggregation in column generation models for resource-constrained covering problems. INFORMS Journal on Computing **29**(1), 170–184 (2017)
46. Pratt, J.W.: Risk aversion in the small and in the large. Econometrica: Journal of the Econometric Society **32**(1/2), 122–136 (1964)
47. Ruszczyński, A.: On convergence of an augmented lagrangian decomposition method for sparse convex optimization. Mathematics of Operations Research **20**(3), 634–656 (1995)
48. Sadykov, R., Lazarev, A., Shiryaev, V., Stratonnikov, A.: Solving a freight railcar flow problem arising in russia. In: ATMOS-13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems-2013. Dagstuhl Open Access Series in Informatics (2013)
49. Sadykov, R., Vanderbeck, F.: Column generation for extended formulations. EURO Journal on Computational Optimization **1**(1-2), 81–115 (2013)
50. Song, Y., Luedtke, J.: An adaptive partition-based approach for solving two-stage stochastic programs with fixed recourse. SIAM Journal on Optimization **25**(3), 1344–1367 (2015)
51. Sponsel, J., Dür, M.: Factorization and cutting planes for completely positive matrices by copositive projection. Mathematical Programming **143**(1-2), 211–229 (2014)
52. Sun, Y., Andersen, M.S., Vandenberghe, L.: Decomposition in conic optimization with partially separable structure. SIAM Journal on Optimization **24**(2), 873–897 (2014)
53. Vandenberghe, L., Andersen, M.S.: Chordal graphs and semidefinite optimization. Foundations and Trends in Optimization **1**(4), 241–433 (2015)
54. Vielma, J.P., Ahmed, S., Nemhauser, G.L.: A lifted linear programming branch-and-bound algorithm for mixed-integer conic quadratic programs. INFORMS Journal on Computing **20**(3), 438–450 (2008)
55. Von Hohenbalken, B.: Simplicial decomposition in nonlinear programming algorithms. Mathematical Programming **13**(1), 49–68 (1977)
56. Wachter, A., Biegler, L.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Mathematical Programming **106**(1), 25–57 (2006)
57. Wang, J., Ralphs, T.: Computational experience with hypergraph-based methods for automatic decomposition in discrete optimization. In: International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems, pp. 394–402. Springer (2013)
58. Wang, Y., Yin, W., Zeng, J.: Global convergence of admm in nonconvex nonsmooth optimization. Journal of Scientific Computing **78**(1), 29–63 (2019)
59. Zangwill, W.I.: The convex simplex method. Management Science **14**(3), 221–238 (1967)
60. Zheng, Y., Fantuzzi, G., Papachristodoulou, A., Goulart, P., Wynn, A.: Fast admm for semidefinite programs with chordal sparsity. In: 2017 American Control Conference (ACC), pp. 3335–3340. IEEE (2017)
61. Zheng, Y., Fantuzzi, G., Papachristodoulou, A., Goulart, P., Wynn, A.: Chordal decomposition in operator-splitting methods for sparse semidefinite programs. Mathematical Programming **180**(1), 489–532 (2020)