HAL
open science

Uncertainties in Optimization

Marie-Liesse Cauwet

To cite this version:

Marie-Liesse Cauwet. Uncertainties in Optimization. Optimization and Control [math.OC]. Université Paris Saclay (COmUE), 2016. English. NNT : 2016SACLS308 . tel-01422274

HAL Id: tel-01422274

https://hal.science/tel-01422274v1

Submitted on 24 Dec 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

NNT : 2016SACLS308

Thèse de doctorat de l'Université Paris-Saclay préparée à l'Université Paris-Sud

Ecole doctorale n°580
Sciences et Technologies de l'Information et de la Communication
Spécialité de doctorat : Informatique

par

Mme Marie-Liesse Cauwet

Traitement de l'Incertitude en Optimisation

Thèse présentée et soutenue à Orsay, le 30 septembre 2016.

Composition du Jury :

M.	Vianney Perchet	Professeur CMLA ENS Paris-Saclay	(Président du jury)
M.	Dirk Arnold	Professeur Dalhousie University	(Rapporteur)
M.	Thomas Jansen	Senior Lecturer Aberystwyth University	(Rapporteur)
M.	Sylvain Arlot	Professeur Université Paris-Sud	(Examinateur)
Mme	Emilie Kaufmann	Chargée de recherche CNRS CRIStAL	(Examinatrice)
M.	Louis Wehenkel	Professeur Université de Liège	(Examinateur)
M.	Marc Schoenhauer	Directeur de recherche INRIA, Université Paris-Sud	(Co-encadrant de thèse)
M.	Olivier Teytaud	Directeur de recherche INRIA, Université Paris-Sud	(Directeur de thèse)

Abstract

This research is motivated by the need to find out new methods to optimize a power system. In this field, traditional management and investment methods are limited when confronted with highly stochastic problems which occur when introducing renewable energies at a large scale. After introducing the various facets of power system optimization, we discuss the continuous black-box noisy optimization problem and then some noisy cases with extra features.

Regarding the contribution to continuous black-box noisy optimization, we are interested into finding lower and upper bounds on the rate of convergence of various families of algorithms. We study the convergence of comparison-based algorithms, including Evolution Strategies, when confronted with different strengths of noise (small, moderate and big). We also extend the convergence results in the case of value-based algorithms when dealing with small noise. Last, we propose a selection tool to choose, between several noisy optimization algorithms, the best one on a given problem.

For the contribution to noisy cases with additional constraints, the delicate cases, we introduce concepts from reinforcement learning, decision theory and statistic fields. We aim to propose optimization methods which are closer to reality (in terms of modelling) and more robust. We also look for less conservative power system reliability criteria.

Résumé

Ces recherches s'inscrivent dans la nécessité de développer de nouvelles méthodes d'optimisation des systèmes électriques. Optimisation du contrôle d'un système électrique, mais aussi optimisation des investissements à réaliser: nouvelles connexions, unités de production et nouveaux stockages. En particulier, l'introduction massive d'énergies renouvelables rend caduques les méthodes usuelles de contrôle qui ne traitent pas efficacement le caractère fortement aléatoire de la production (éolien, solaire).

Le chapitre 1 introduit les enjeux de la transition énergétique: enjeux environnementaux, politiques et technologiques. Le problème d'optimisation d'un système électrique est formalisé sous la forme d'une équation à 5 variables: les investissements, les incertitudes non stochastiques telles que les incertitudes géopolitiques et technologiques, les politiques de contrôle, les configurations du réseau et les incertitudes stochastiques telles que la demande en électricité, la production éolienne ou l'arrivée d'eau dans un barrage pour un jour donné. Les méthodes les plus classiques d'optimisation sont ensuite présentées et discutées. 4 questions de recherche, développées et étudiées dans cette thèse, sont formulées:

- Question de recherche #1: Comment résoudre le problème d'Unit Commitment sans supposer la linéarité ou la convexité de la fonction de coût et sans simplifier excessivement le processus aléatoire ?
- Question de recherche #2: Comment approximer, en un temps de calcul raisonnable, un équilibre de Nash dans un cadre stochastique ?
- Question de recherche #3: Comment réduire le biais induit par l'utilisation d'un échantillon de taille modeste dans un problème de planification de développement de capacités ?
- Question de recherche #4: Quels sont les taux de convergence optimaux de différentes familles d'algorithmes d'optimisation continue dans un contexte bruité et boîte noire ?

La première partie est dévolue à la question de recherche #4. Le chapitre 2 commence par définir l'optimisation continue dans un contexte bruité et boîte noire et dresse la revue de littérature. En particulier, une partition est faite entre deux grands types d'algorithmes: les algorithmes basés sur les évaluations et les algorithmes basés sur les comparaisons. Le chapitre 3 étudie le comportant des stratégies d'évolutions, un cas particulier d'algorithmes basés sur les comparaisons, face à un bruit faible: au prix d'un petit nombre de réévaluations, cette famille d'algorithme converge aussi vite que dans le cas non bruité. Le chapitre 4 montre, dans le cas d'un bruit fort (additif), une borne inférieure sur le taux de convergence[1] des algorithmes basés sur les comparaisons lorsqu'ils échantillonnent toujours dans un domaine restreint, près de l'approximation courante de l'optimum. Sous ces hypothèses, le taux de convergence n'est jamais plus petit que $-1/2$, alors que les algorithmes basés sur les évaluations atteignent un taux de -1. Le chapitre 5 prouve que, néanmoins, les algorithmes basés sur les comparaisons peuvent égaler ceux basés sur les évaluations en terme de taux de convergence, si l'on effectue des ré-échantillonnages 'loin' de l'optimum, toujours dans le cas d'un bruit additif. Le chapitre 6 s'intéresse aux algorithmes basés sur les évaluations, et propose l'étude simultanée d'un bruit faible, modéré et fort. On retrouve les résultats déjà connus dans le cas d'un bruit fort, et des nouvelles bornes sont démontrées pour les autres types de bruit. Le chapitre 7 propose une méthode de sélection d'algorithmes dans le cadre bruité: sélection d'un type d'algorithme (basé sur des évaluations ou des comparaisons) et sélection de la paramétrisation d'un algorithme. Pour cela, on considère un portfolio qui, notablement, sélectionne un algorithme en se basant sur les itérations *passées*, et non pas courantes. Cette subtilité est essentielle pour obtenir une convergence; il est prouvé que le portfolio sélectionne alors presque toujours un des algorithmes les plus performants.

La seconde partie regroupe les cas bruités plus délicats. Le chapitre 8 traite la question de recherche #1, où le problème est bruité *et* dépend du temps. Ce chapitre présente et étudie une méthode qui permet de mieux prendre en compte des coûts non linéaires et non convexes, ainsi qu'un processus stochastique non markovien. Une étude théorique démontre que cette méthode peut atteindre une politique optimale et plusieurs expériences (sur un problème de gestion de multiples batteries et deux problèmes hydroélectriques) montrent ses bons résultats en pratique. Le chapitre 9 adresse la question de recherche #2, i.e. optimisation bruitée *et* adversariale. Plusieurs variantes de l'algorithme de Grigoriadis

[1] i.e. une borne supérieure sur la vitesse maximale de convergence.

et Khachiyan adaptées au cas bruité sont proposées et étudiées. Le chapitre 10 porte sur la question de recherche #3, autour de l'optimisation bruitée *et* à taille d'échantillon réduite. Des méthodes de bootstrap et de validation croisée sont utilisées sur un cas test artificiel, permettant de réduire le biais issu du petit nombre d'échantillons collectés.

La troisième partie conclut et met l'accent sur des points intéressants à approfondir lors de futures recherches.

Acknowledgement

Ph.D. studies are the result of rigour, patience and passion, but it can not happen without the help and advices of experienced professors and researchers and the support and assistance of family and friends.

First and foremost, I would like to thank my advisor Olivier Teytaud, who gave me the opportunity to pursue doctoral studies. Not only did he introduce me to the research world but his kindness and patience in daily life make him also a role model for me.

I am very grateful to Dirk Arnold and Thomas Jansen who kindly accepted taking time to review my Ph.D. thesis and provided great comments and appreciation of this work. My sincere thanks also go to the rest of the thesis committee, Sylvain Arlot, Emilie Kaufmann, Vianney Perchet, Marc Schoenhauer and Louis Wehenkel. Their insightful comments and hard questions prompted me to widen my research from various perspectives.

Thanks a lot to Marc Schoenhauer and Michèle Sebag for their assistance on the finishing straight line of this 3 years journey.

Many thanks to the administrative staff of INRIA and of the Université Paris-Sud, Olga Mwana Mobulakani, Stéphanie Druetta, for their invaluable assistance and Marie-Carol Lopes, for her help with the financial side of my Ph.D., funded by the Post project (Pia-Ademe).

I can not imagine better conditions to conduct my researchs than within the TAO team, its relaxed but working atmosphere. Especially thanks to my fellow labmates: Adrien (for welcoming me first in TAO then in Université de Liège), Ahmed (for keeping my plant alive), Alexandre (for his cool board games), Anne, Antoine (for his generosity), Arthur, Asma, Aurélien, Basile, Baptiste (for the fun we had in Taiwan), Constance, Corentin, David, Diviyan, Edgar (for his help with my English writing skills), Gaétan (for his wisdom), Guillaume, Jean-Baptiste (for his good advices at the very beginning of my Ph.D.), Jean-Joseph, Jérémie (for his programming expertise), Jérémy (for sharing with me his defense experience),

Jialin (for her joy of life and humour), Karima, Lovro, Luigi (for his delicious cooking), Manuel, Marco (for the coffee machine maintenance), Marta, Nacim, Nicolas (for feeding me and for his planning skills), Odalric (for our inspirational talks to make the world a better place), Philippe, Pierre-Yves (for his kindness), Reben, Riad (for his silly jokes), Sandra (for her unfailing support during our late deadlines), Thomas (for his priceless recipe of "poulet à la Thomas"), Victor, Vincent, Wassim (for our high-quality linguistic debates). Some of them are beginning or finishing a Ph.D., some of them are already experienced researchers, I wish you all the very best!

I got the chance to develop some fruitful cooperation within Taiwanese students and professors, a part being funded by the Research and Practical Training Program in Taiwan 2014 launched by the National Science Council of Taiwan. I am very thankful to Prof. Chang-Shing Lee, Mei-Hui Wang and Prof. Shi-Jim Yen for the warm welcoming I had in the National University of Tainan and in National Dong Hwa University. A special mention to Ching-Nung, Chi-Shiang, Jian-Lin, Kuan-Yi, Ting-Tzu, Tsang-Cheng and Sarah who made me discover Taiwanese food, culture and way of life.

I also really appreciated working with Amélie, Johanne and Peio on various projects. Thanks to all my former professors, who gave me a taste for research and a sense of rigour. Thanks to all my friends of master, with a special mention to Anthony, Benjamin, Jhih-Huang, Mélanie, Tiago, Valérie, Vincent, Yann and Yunlong. Thanks to Efthymios and Laurine for their kind welcome in Université de Liège.

Thanks to the members of my family for their continuous support. Many thanks to my mother and my aunt Martine, who made the best ever "pot de thèse", with these awesome "spécialités tourangelles et auvergnates" that I strongly recommend to my foreigner friends. Thank you for being there when I needed you.

Thanks to Fedor, for always being on my side. Since I know you, I jumped from a plane, rode a horse and survived hard core hiking in the wild Altai mountains. You are the best.

Thank to all of you who helped me directly or indirectly during my Ph.D. studies and that I did not mentioned above. Writing some acknowledgements is always a tough exercise, my deepest fear being to forget someone. So I apologize in advance if it is the case, and I invite the potential aggrieved person to drink a coffee, tea or beer.

Contents

1 A beautiful application: power systems **19**
1.1 Motivations 19
1.2 Optimization in power systems 22
1.2.1 Parameters of the cost function 22
1.2.2 Optimization problem 25
1.2.3 How to handle this? 26
1.3 Structure of the thesis & contributions 33
1.3.1 Contribution in noisy AND black-box continuous optimization 33
1.3.2 Delicate cases 33

Part I Contributions in Continuous Black-Box Noisy Optimization **36**

2 Background review **37**
2.1 The different flavours of optimization 37
2.1.1 White, gray or black? 37
2.1.2 Global or local? 38
2.1.3 Noisy or noise-free? 38
2.2 Convergence(s): what makes an algorithm 'good'? 43
2.2.1 Exploration vs. exploitation 43
2.2.2 Convergence criteria 44
2.2.3 Type of convergence 46
2.2.4 Discussion 48

2.3 Algorithms and state of the art 49
2.3.1 Value-based algorithms 49
2.3.2 Comparison-based algorithms 57
2.3.3 General lower bound 60
2.4 Contributions in noisy black-box continuous optimization 63
2.4.1 Comparison-based algorithms 63
2.4.2 Value-based algorithms and small noise 63
2.4.3 Right algorithm? Right parameters? 64

3 Evolution Strategies confronted with small noise 65
3.1 Log-linearity 66
3.1.1 Preliminary: noise-free case 66
3.1.2 Noisy case 67
3.2 Experiments: how to choose the right number of resampling? . . . 72
3.3 Conclusion 73

4 Convergence rate of Evolution Strategies with additive noise: a lower bound 77
4.1 Theoretical analysis 78
4.1.1 Formalization of algorithms 78
4.1.2 Lower bound for Simple Evolution Strategies 82
4.2 Experimental verification of the lower bound 84
4.2.1 Fast convergence: Shamir's algorithm 84
4.2.2 Slow convergence: UH-CMA-ES and (1+1)-ES 86
4.3 Conclusion 87

5 Comparison-based algorithms can be fast! 92
5.1 Comparison procedure 93
5.2 Sphere function 94
5.2.1 In dimension 1 94
5.2.2 Multidimensional sphere function 96
5.3 General quadratic forms 97
5.4 Experiments 102
5.5 Conclusion 102

6 Newton's method: upper bounds 105
6.1 The Iterative Noisy Optimization Algorithm (INOA) 107
6.1.1 General framework 107

6.1.2 Examples of algorithms verifying the Low-Squared Error (LSE) assumption . . . 109
6.2 Convergence rates of INOA . . . 115
6.2.1 Rates for various noise models . . . 115
6.2.2 Application: the general case . . . 117
6.2.3 Application: the smooth case . . . 118
6.3 Conclusion . . . 118

7 Algorithm Portfolio 122
7.1 Algorithm selection . . . 123
7.2 Algorithms . . . 126
7.2.1 Notations . . . 126
7.2.2 Definitions and criteria . . . 126
7.2.3 Portfolio algorithms . . . 127
7.3 Analysis . . . 130
7.4 Conclusion . . . 141

Part II Contributions to Delicate Cases 143

8 A Consistent Model Predictive Control 144
8.1 Introduction . . . 145
8.1.1 Formalism of Markov Decision Processes . . . 145
8.1.2 State of the art in dynamic optimization . . . 146
8.2 Direct Model Predictive Control (DMPC) . . . 147
8.2.1 DMPC: Formulation . . . 147
8.2.2 DMPC brings consistency into MPC . . . 148
8.3 Consistency analysis . . . 148
8.3.1 Optimality of DMPC . . . 149
8.3.2 Nonlinear Setting . . . 152
8.4 Experiments . . . 153
8.4.1 Experiments with a 10 batteries problem . . . 153
8.4.2 Experiments on a real-world hydroelectric problem . . . 154
8.5 Conclusion . . . 156

9 Stochastic zero-sum games 159

9.1 Introduction 160
9.1.1 Adversarial Matrix Games & Nash Equilibrium 160
9.1.2 Outline 162
9.2 Algorithms and settings 162
9.2.1 Settings & motivations 162
9.2.2 Algorithms 164
9.3 Theory 164
9.3.1 Extension of Grigorias and Khachiyan's result to setting 2 165
9.3.2 Another algorithm with K^2 processors: `ParaNash` 167
9.4 Experiments 170
9.4.1 Randomly uniformly drawn Bernoulli parameters 170
9.4.2 Other distributions 172
9.4.3 Experiments on the Pokémon problem 173
9.5 Conclusion 173

10 Multivariate bias reduction in capacity expansion planning 180
10.1 Optimization of power systems capacities: Sample Average Approximation & bias 180
10.2 M-estimators and bias 181
10.2.1 Sample Average Approximation (SAA) 181
10.2.2 Bias & Simple Regret 182
10.3 Bias reduction 183
10.3.1 Resampling estimates for bias reduction 183
10.3.2 Dimension reduction for bias reduction 184
10.4 Model selection 185
10.4.1 Leave-k-out (Lk) for model selection 186
10.4.2 Penalized-Cross-Validation for model selection (pen$k$-F) . 186
10.4.3 Meta-estimate using model selection 187
10.4.4 The margin method 187
10.5 Experiments 188
10.5.1 Test case 188
10.5.2 Estimators 189
10.5.3 Experimental results 189
10.6 Conclusion 195

Part III Summary, Discussion and Perspectives 200

11 Summary, Discussion, Perspectives 201
11.1 Part I: Summary and Discussion 201
11.1.1 Summary 201
11.1.2 Perspectives & discussion 203
11.2 Part II: Summary and Discussion 206
11.2.1 Summary 206
11.2.2 Perspectives & discussion 207
11.3 Reflections on power systems optimization 208

Appendix A Summary of Notations 211

Appendix B Lower bound in $\Omega(n^{-1})$ or $\Omega(n^{-1/2})$: a discussion 213
B.1 Polyak-Tsybakov vs. Shamir 213
B.2 Asymptotic vs. non asymptotic setting 215
B.3 Illustration 217

Appendix C Proofs of Sections 6.1.2 and 6.2.2 in Chapter 6 219

Appendix D Appendix: Proof of Lemma 8.3.1 224

Bibliography 226

List of Figures

1.1 Cutting plane method. 27

3.1 Self Adaptive Evolution Strategy: Median results 74
3.2 Self Adaptive Evolution Strategy: Mean results 75

4.1 Shamir's algorithm on the noisy sphere function. 86
4.2 UH-CMA-ES algorithm on the noisy sphere function. 87
4.3 $(1+1)$ Evolution Strategy with resamplings on the noisy sphere function. 88

5.1 COPQUAD when confronted with a strong noise. 103
5.2 COPQUAD when confronted with a small noise. 104

8.1 DMPC vs. MPC on a 10 batteries problem. 157
8.2 DMPC vs. MPC on a real-world hydroelectric problem. 158

9.1 Comparison of algorithms computing the Nash equilibrium (Bernoulli parameter in $(0,1)$). 171
9.2 Comparison of algorithms computing the Nash equilibrium (Bernoulli parameter in $(.45,.55)$). 172
9.3 Comparison of algorithms computing the Nash equilibrium (Pokemon game). 174

10.1 Comparison of bias reduction methods when confronted with a strong penalty. 190
10.2 Comparison of bias reduction methods when confronted with a small penalty. 191
10.3 Comparison of bias reduction methods when confronted with a moderate penalty. 192
10.4 Experiments on Penalized-Cross-Validation. 196
10.5 Experiments on Penalized-Cross-Validation with margin method. . 197

B.1 Experiments on Polyak-Tsybakov's algorithm 218

List of Tables

2.1 Upper bound on the *SR* using Polyak-Tsybakov's algorithm. . . . 56

6.1 Convergence rates of INOA when confronted with various strengths of noise. 119

10.1 Estimators compared in the experiments. 193
10.2 Performance of Penalized-Cross-Validation (CV) on various values of the penalty 195

List of Algorithms

2.1 $(1,\lambda)$-Self Adaptive Evolution Strategy. 58
3.1 A general framework for Evolution Strategies with constant number of resamplings. 69
3.2 (μ,λ)-Self Adaptive Evolution Strategy with constant number of resamplings. 76
4.1 General optimization framework. 79
4.2 Shamir's algorithm (written in the general optimization framework). 85
4.3 $(1+1)$ Evolution Strategy with resamplings. 89
4.4 Improved $(1+1)$ Evolution Strategy with resamplings. 91
5.1 Comparison procedure for sphere function in dimension 1 (COPS1). 94
5.2 Comparison procedure for the sphere function (COPS). 97
5.3 Comparison procedure for quadratic functions (COPQUAD). . . . 98
6.1 Iterative Noisy Optimization Algorithm (INOA). 108
7.1 Noisy Optimization Portfolio Algorithm (NOPA). 128
7.2 Improved Noisy Optimization Portfolio Algorithm (INOPA). . . . 131
9.1 Grigoriadis and Khachiyan's algorithm. 176
9.2 NaiveParaGrigoriadis. 177
9.3 ParaGrigoriadis. 178
9.4 ParaNash. 179

Abbreviations

a.s. almost surely, i.e. with probability 1.

e.g. exempli gratia, means “for example” in Latin.

i.e. id est, means “that is (to say)” or “that means” in Latin.

s.t. such that.

w.l.o.g without loss of generality.

Acronyms

ASR Approximate Simple Regret. 42

BS Bootstrap. 29, 180, 181, 186, 190, 195

CLOP Confident Local Optimization. 39, 53

CMA-ES Covariance Matrix Adaptation - Evolution Strategy. 56

CR Cumulative Regret. 42, 44–46, 49, 59, 80, 103, 104, 112, 114–117, 157, 199, 201, 202, 209

CSA-ES Cumulative Step-size Adaptation - Evolution Strategy. 56, 64

CV Cross-Validation. 11, 13, 14, 29, 182–186, 191–195, 204, 205

DMPC Direct Model Predictive Control. 10, 13, 31, 141, 142, 144–146, 148–153, 199, 203–205

DPS Direct Policy Search. 26, 29, 38, 141, 144, 153, 198

ERM Empirical Risk Minimizer. 179, 180, 182, 184, 186, 190–194

ES Evolution Strategy. 9, 13, 15, 30, 37, 41, 54–57, 60–66, 69, 70, 73–75, 77–79, 81, 83–87, 102, 119, 129, 199–202

INOA Iterative Noisy Optimization Algorithm. 9, 10, 15, 104, 105, 112–116

INOPA Improved Noisy Optimization Portfolio Algorithm. 15, 121, 122, 126–128, 134, 135, 138, 139

JK Jackknife. 29, 180, 181, 184, 186, 190–192, 195

LSE Low-Squared Error. 10, 104–108, 110–115, 117, 118

LTM Long Term Management. 151, 152, 155

MDP Markov Decision Process. 10, 25, 26, 141, 142, 149

MPC Model Predictive Control. 10, 13, 26, 31, 141–145, 151–153, 203

NOPA Noisy Optimization Portfolio Algorithm. 15, 121, 122, 124, 125, 127, 131, 134–136, 138, 139

R-EDA Racing-based Estimation of Distribution Algorithms.

SAA Sample Average Approximation. 11, 22, 32, 177–179, 195

SA-ES Self Adaptive Evolution Strategy. 13, 15, 55, 64, 71, 72, 154

SDDP Stochastic Dual Dynamic Programming. 25, 26, 141, 143–145, 205

SDP Stochastic Dynamic Programming. 26, 141, 143, 145

SR Simple Regret. 11, 14, 41, 42, 44, 45, 47–53, 56, 57, 60, 74, 81, 87, 89–91, 93, 94, 102–104, 106, 112, 114–117, 123, 124, 127, 129, 131, 133, 135, 137, 157, 179, 180, 184–186, 192, 193, 199, 201, 202, 209, 210, 213–215

UH-CMA-ES Uncertainty Handling - Covariance Matrix Adaptation - Evolution Strategy. 9, 13, 56, 57, 83, 84, 87

UR Uniform Rate. 41, 43–46, 54, 57, 103, 104, 115, 117, 209

Chapter 1

A beautiful application: power systems

1.1 Motivations

Energy concerns are already in the spotlight since a few years. Opening your favourite newspaper or social network, chances are good for you to find an article related to energy transition.

Environmental issue & climate change. Fossil energies - oil, coal and natural gas - contribute for a large part to premature deaths across the world, up to 8 millions deaths per year according to [World Health Organization, 2015], notably through air pollution; see also [Conca, 2012] for an interesting discussion on the rate of deaths per source of energy. Along with the air pollution, public opinion worldwide worries about the use of nuclear energy which raises the matters of long time storage of the wastes, accidents or terrorist threat. Energy production is also by far the first source of greenhouse gas emission [International Energy Agency, 2015].

Economic weight. Economy and energy are intrinsically linked [Stern and Enflo, 2013], think e.g. about the energy crisis of the 1970s. In particular, [Giraud, 2015] discussed the correlation between GDP and energy. A dynamic economy relies heavily on the access to a cheap and abundant source of energy [Fonteneau, 2015].

Political response. Confronted with these challenges, we encounter local responses - at the scale of a country - such as fixing the electricity price, subsiding a given energy source, looking for energetic independence or allowing or not shale gas exploitation. Some structures are in charge of the studies and development of concrete new solutions, such as the ADEME[1] in France, which gave me the opportunity to work on this topic through the Post project[2]. At a regional level, a group of countries can agree on some common policy such as the so-called $(N-1)$ *standard*[3], voted by the European parliament [Official Journal of the European Union, 2010]. Global decisions are also discussed within international conferences, resulting into some accords - the last one being the *Paris Agreement*, where the members agreed to try to reduce their greenhouse gas emissions in order to not exceed 2°C of global warming [United Nations, 2015]. On the other hand, this resolution will be difficult to follow since the demand for energy keeps growing, notably from the emerging countries, and could increase by 37% by 2040 according to [International Energy Agency, 2014].

With new needs come new technologies. Renewable energies are significantly cleaner - in terms of CO_2 and particulate matter - than fossil energy [International Energy Agency, 2015, Ohlström et al., 2000]. However, apart from specific sources such as hydroelectricity, they are highly stochastic and difficult to forecast. Even with a small percentage of renewable energy involved in the power network, problems arise in case of sudden changes (e.g. in luminosity). Thus, solar production is impacted very quickly and globally by an eclipse [SolarPower Europe, 2015]. Through domino effect, such phenomenon might cause a global black-out without a careful and coordinated planning of the different electrical grid's operators, since such an event is particularly easy to predict.

The massive introduction of renewable energy implies changes in the design of the power grid, e.g. additional storage capacities and use of *smart grids*, able to drive accurately the consumption. In particular, it has been pointed out that the extensive use of renewable energy implies a paradigm shift: instead of adapting the production to the demand - as it is done nowadays - the demand should be smoothed in order to fit the current production [Marchal, 2015]. Large parks of

[1] `http://www.ademe.fr/en`

[2] `http://www.post.artelys.com/`

[3] This standard requires each member state to be able to tackle an unexpected outage of their single largest piece of gas infrastructure, i.e., satisfying the demand with the remaining network [Ralf et al., 2014].

electric vehicles could support this smoothing via a smart control of the batteries loading. Space smoothing could also be applied thanks to long distance connections (High Voltage Direct Current - HVDC); see also the ambitious *global grid* concept [Ernst, 2015b, Chatzivasileiadis et al., 2013]. On the contrary, *microgrids*, small networks of the size of one house or a few streets, are also under study. Microgrids are composed of various renewable energy sources and a battery, possibly completed with a diesel generator [Ernst, 2015a] - they might be or not be connected to the grid.

Where does computer science stand in all of this? As [MacKay, 2009] says, "we need numbers, not adjectives." And these numbers should be reliable and meaningful.

Simulating and optimizing power systems is crucial for testing the validity and cost of some scenarios:

- What are the costs (economical, ecological) of a purely renewable system?
- Consider a limited budget (bound on investments) over the next 50 years: what is the best investment planning?
- What is the ecological/economical benefit, if we can relax the constraint of national independence?
- What is the impact of a given gas supply cut-off / what is the best adaptation strategy to such a gas supply interruption?

An important task is to build reliable power system modelling tools. Big power system companies use such modelling platforms but they are not available in open source. It is possible to find some platforms such as the Artelys Crystal Super Grid[4] and some libraries, such as Simscape Power Systems in Matlab[5] or Modelica PowerSystems library[6].

Along with modelling, we need detailed data, either archive or simulation. When wind and solar power are involved in a power grid, we need time and space series in order to forecast accurately the wind speed and daylight, especially we need to measure their correlation. Some benchmarks have been developed in order

[4] `https://www.artelys.com/en/applications/artelys-crystal-energy/artelys-supergrid`

[5] `http://fr.mathworks.com/products/simpower/`

[6] `https://github.com/modelica/PowerSystems`

to test and compare various optimization algorithms, such as the COCO/BBOB platform[7] for continuous optimization or OpenAI[8] for reinforcement learning, see also [Castronovo et al., 2016].

Once we have both tools at hand, i.e. power system modelling and data, we want to *optimize* this system. The subject of this thesis is precisely this optimization side. Especially, as it will be discussed below, the increase in volatility production requires new models of energy management. We study optimization processes able to handle stochastic effects.

1.2 Optimization in power systems

A power grid consists of a transmission network, a distribution network, loads and power plants. Optimizing this power system means optimizing a given cost function under *constraints*. The cost function includes economical costs, maintenance costs and environmental costs. Economical costs take into account risks of failure [Autorita per l'Energia Elettrica e il Gas, 2004] and maintenance costs incorporate risks for workers. The constraints are operational constraints of power systems and demand satisfaction.

We propose through Eq. 1.2 an optimization problem for power systems. We present the motivations behind the variables choices involved in this equation. Especially, we do not detail the modelling aspect, i.e., the physical laws such as the Kirchhoff law, which govern the current and voltage of an electrical network, but give some references for the interested reader. We focus on the optimization aspect. We review the different optimization solutions at hand and the challenging facets arising, such as high dimensionality, stochasticity, non-convexity, non-linearity or non-Markovianity. We aim to study and develop principled methods able to deal with such problems - thus our contribution is mainly theoretical and the scope of this thesis is not exclusive to power systems, but might be useful in every application dealing with similar difficulties.

1.2.1 Parameters of the cost function

The cost function depends on investments, non stochastic uncertainties, management policy (unit commitment), network configuration, and 'real' stochastic uncertainties. We detail below these parameters.

[7] `http://coco.gforge.inria.fr/`

[8] `https://openai.com/blog/openai-gym-beta/`

Investments $\mathscr{I}$.

Investments regard the new capacities - storages, connections, plants - that should be added to the power grid. Where should they be added? Which kind of capacities: HVDC connection or not, storage, solar plant, wind plant, thermal plant, etc...? What dimension/size is optimal? Quantifying the optimal connection capacities and storage capacities at the scale of a continent or more is an important optimization problem, with budget in dozens or hundreds of billions of euros. There are high level facts which are well known: in the European grid, conditions are better for wind power in the north, for solar power in the south, for additional hydroelectric storage in Scandinavia. Also, Africa is not that far - there are already connections between Europe and Africa [SYSTINT Workgroup, 2007], and increasing these connections is a possibility.

Non-stochastic uncertainties $\mathscr{U}$.

In the setting of long term planning, i.e. a time horizon of several decades, non-stochastic uncertainties must be included. It encompasses uncertainties which can not be modelled, such as

- political uncertainties: real CO_2 penalization or not, cut off of Russian gas exportation, energy sources subsidies, oil prices;
- technological uncertainties: solar and wind plants efficiency, power to gas efficiency, large use of electric cars or not;
- climate change uncertainty: how many degrees of global warming?

As a first approximation, it seems reasonable to assume that everyone collaborates: countries of the same geographical area develop a common power grid such that the costs (economical, environmental, ...) are minimum; in this case, the optimization in Eq. 1.2 over $\mathscr{U}$ would be a minimum. However, it often occurs that countries or areas want to have some autonomy in case of problems: we set a maximum over $\mathscr{U}$ in Eq. 1.2, so that we pessimistically consider the worst case.

Management policy $\mathscr{P}_1$.

Given a power network, the energy producer needs to satisfy the demand. This is the management policy, or unit commitment [Padhy, 2004, Sheble and Fahd, 1994]. This problem involves deciding which power plants are

switched on/off and the dispatch, i.e., deciding the power output for each plant. On the power grid, different power plants are available: thermal plants, nuclear power plants, hydroelectric power stations (dams), solar power plants, wind turbines, etc... Each plants needs to satisfy some constraints such as maximum ramping rate, stock management constraints, start up costs, minimum water flows (see e.g. [Bertsimas et al., 2013]). The details of the unit commitment problem are beyond the scope of this thesis, we refer to [Decock, 2014] for more information regarding the constraints and specific costs. We invite the reader interested in hydrothermal scheduling to read more in [Couëtoux, 2013]. The unit commitment problem is a *sequential decision making problem*, also termed *multistage optimization problem* or *dynamic programs*, see [Bertsekas, 1995, Powell, 2007].

Network configuration $\mathscr{P}_2$.

In regard to short term management, security and costs concerns invite to deal with the network reconfiguration. In case of line outage (cf. the $N-1$ standard in Section 1.1), it is cheaper and faster to change the topology of the network, i.e., to switch the transmission line, rather than re-dispatching the power output for each plant in order to avoid overloading of lines. Afterwards, the new topology can be reversed if the line failure is fixed or after a generator dispatch. This problem involves some load flow feasible constraints (e.g. the flow should satisfy the Kirchhoff law and not exceed the line capacity) and is highly relevant in the context of terrorism threat [The Telegraph, 2015]. Furthermore, with the increasing integration of renewable energy- more likely to cause sharp load changes - comes the need to increase the flexibility and efficiency of the power grid. This topic is beyond the scope of this thesis, see [Li et al., 2012, Kezunovic et al., 2014, Hedman et al., 2011, Zaoui et al., 2005] for more details.

Stochastic uncertainties $\omega \sim \Pi$.

The stochastic uncertainties designate the random variables which are generated by an underlying (possibly unknown) probability distribution function Π. In particular, it does not depend on human decisions such as political uncertainties. In the context of energy management, it is, typically (non exhaustively), luminosity, wind speed, inflows [Siqueira et al., 2006] in a dam and demand [RTE-ft, 2014].

We bring the attention to the assumptions on the random variables: are the random variable realizations independent or not? are they Markovian or not? do

they have a finite variance? These assumptions will be discussed throughout this thesis.

The knowledge about this random process varies from one problem to the other. Mainly, we can distinguish between problems where we have a finite sample of random values realizations and those for which we have a generative model available. Between these two extreme settings, one can try to infer the hidden distribution from a sample of realization.

Optimizing against a finite sample of realization is termed *Sample Average Approximation (SAA)*; this is discussed in Chapter 10.

When a generative model is at hand, we consider several strategies. A first approximation is to optimize the cost function over the expectation of the random process, i.e. replace ω with $\mathbb{E}_{\omega\sim\Pi}(\omega)$ in Eq. 1.2. It amounts to performing deterministic optimization, which is already very challenging since power system optimization involves high dimensional problems. As a drawback, this might cost a lot in case of extreme events - drought, heatwave, very cold winter. We aim at encompassing the whole stochastic aspect. That is, we want to optimize taking into account the random process. In this case, we want to optimize the expected costs, computed with respect to the probability distribution of the random process, as in Eq 1.2.

1.2.2 Optimization problem

Consider a function, denoted by COST. It is a measure, economical, ecological, social, of the efficiency of the power system. This function results from some physical laws and/or is given by some experts; see [Decock, 2014, Couëtoux, 2013] for some examples. Consider the set of variables introduced above: investments $\mathscr{I}$, non stochastic uncertainties $\mathscr{U}$, management policy $\mathscr{P}_1$, network configuration $\mathscr{P}_2$ and stochastic uncertainties ω following an unknown probability distribution Π. Then, the following equation can be used to choose the suitable investments:

$$i^* = \arg\min_{i\in\mathscr{I}} \mathbb{E}_{\omega\sim\Pi} \max_{u\in\mathscr{U}} \min_{p\in\mathscr{P}_1} \min_{p'\in\mathscr{P}_2} \textsc{Cost}(i,u,p,p',\omega). \tag{1.1}$$

However, to compute the min over $\mathscr{P}_1$ and $\mathscr{P}_2$ in Eq. 1.1, we make the assumption that ω is known. That is, to assume that failures and weather conditions are perfectly forecast. This is an *anticipativity* assumption, which is less than satisfactory in a stochastic setting with limited forecasts. We propose a better

formulation of the optimization problem, without assuming perfect forecasts, as follows:

$$i^* = \arg\min_{i \in \mathscr{I}} \max_{u \in \mathscr{U}} \min_{p \in \mathscr{P}_1} \min_{p' \in \mathscr{P}_2} \mathbb{E}_{\omega \sim \Pi} \, \textsc{Cost}(i, u, p, p', \omega). \tag{1.2}$$

We aim to bring some elements of response to this equation, significantly more difficult to solve than Eq. 1.1.

1.2.3 How to handle this?

As each variable has different features (dimension, domain, constraints,...), each part of Eq. 1.2 is handled differently. We review briefly these methods and discuss their advantages and drawbacks.

Cutting plane method

The *cutting plane method* can be used to optimize the sub-equation:

$$i^* = \arg\min_{i \in \mathscr{I}} \min_{p \in \mathscr{P}_1} \mathbb{E}_{\omega \sim \Pi} \, \textsc{Cost}(i, p, \omega)$$

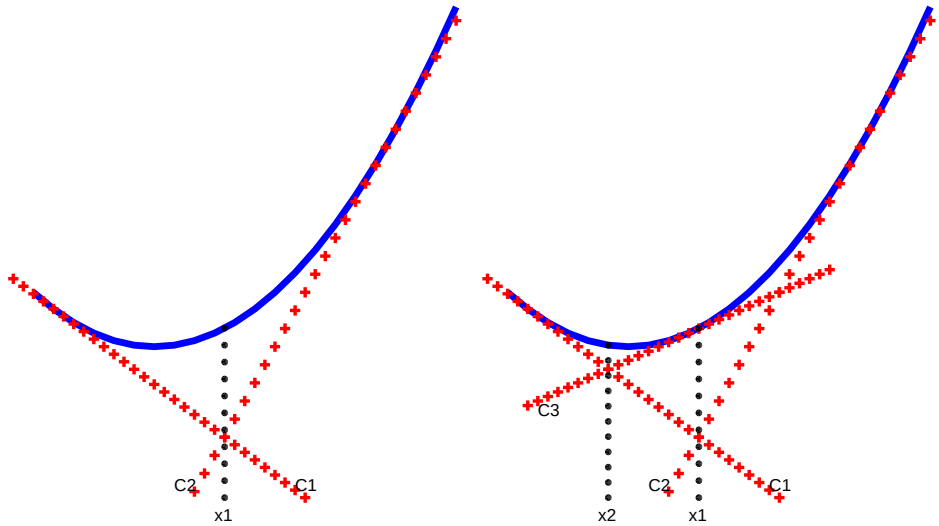
within a moderate computational cost.

The cost function must be convex and have some sub-gradient, but not necessary any gradient. *Kelley's method* [Cheney and Goldstein, 1959, J. E. Kelley, 1960], *bundle method* [de Oliveira and Sagastizàbal, 2014] or *Benders decomposition* [Benders, 1962] are classical variants of the cutting plane method. It consists in approximating the cost function by a convex piecewise linear function APPROX:

$$\forall i \in \mathscr{I}, \quad \textsc{Approx}(i) = \max_{1 \leq j \leq d} l_j(i),$$

where l_j is a linear function $\forall j \in \{1, \ldots, d\}$ and d an integer.

It is assumed that the optimum of APPROX is a good approximation of the optimum of the cost function. The piecewise linear function is obtained by adding at each iteration a new plane, using the sub-gradient of the cost function computed at the current approximation of the optimum (see Fig. 1.1). The strength of this method is that it can be coded in a linear problem: it is solvable in polynomial time [Karmarkar, 1984]. Hence it is very fast in theory and usually solved in a reasonable time in practice.



(a) (b)

Figure 1.1: Cutting plane method. The blue plain curve is the function to optimize. We first compute a given number of planes big enough to be able to have a first approximation, here 2 (C1 and C2 in red '+') in Figure 1.1a. They are computed using the subgradient of the cost function. Their minimum x_1 is the estimate of the optimum of the cost function. Then, at each iteration, we add one cut and update the approximation; Figure 1.1b displays the 2nd iteration.

Example 1.2.1 (Lagrangian relaxation)**.** *Solving the Unit Commitment Problem, i.e. solving the minimization over $\mathscr{P}_1$ in Eq. 1.2 consists in solving a minimization problem under constraints. For handling this, Lagrangian relaxation [Bertsimas and Tsitsiklis, 1997] is a common method. The dual problem obtained is then possibly solved with the cutting plane method. See [Belloni et al., 2003] for a beautiful application of this method to the Brazilian Power System. We point out also that in this case, the Lagrangian multipliers represent the marginal costs of the various power plant under study.*

Example 1.2.2 (Stochastic Dual Dynamic Programming (SDDP))**.** *Stochastic Dual Dynamic Programming*[9] *[Pereira and Pinto, 1991] is another method to tackle the Unit Commitment Problem. In particular, it handles the Markov Decision Processes. The cutting plane method is the central idea behind the SDDP algorithm (used in the backward path). Note that SDDP handles the stochastic uncertainties, but requires convex Bellman values and a moderate complexity of random processes state.*

Nonetheless, convexity assumption is a strong condition, which might not hold in practice when it comes to power system. We give three arguments which support this remark.

- **Hydroelectricity.** The efficiency in the dam depends of the height of fall. Hydroelectricity also implies various turbines with different outputs. This results in some non-convex cost function, see [Couëtoux, 2013].
- **Gas turbine.** Non-convex effects similar to the hydroelectric case might occur in the gas turbine case.
- **Economies of scale.** Generally, costs per kilowatt of capacity decrease as size increases, e.g. for CHP[10] or for solar power [International Renewable Energy Agency, 2012]. This is why we expect non-convex cost functions.

Reinforcement Learning & Control

The unit commitment sub-problem:

$$p^* = \underset{p \in \mathscr{P}_1}{\arg\min}\ \mathbb{E}_{\omega \sim \Pi}\, \textsc{Cost}(p, \omega)$$

can be modelled in the following way.

Given an initial state s_0, a policy p, a transition function $\mathfrak{T}$, a final step time T and a sequence of random variables $\omega_0, \ldots, \omega_{T-1}$, we define:

$$\begin{aligned}
a &= p(s,t)\text{: the decision at } t,\\
s &= \mathfrak{T}(s,t,a,\omega_t)\text{: the state at } t,\\
c_t &= C_t(s,a) \in \mathbb{R}\text{ : the cost at time } t,\\
\textsc{Cost}_p &= \sum_t c_t \in \mathbb{R}\text{ : the total cost function.}
\end{aligned}$$

[9] See Chap. 8 for the unknown vocabulary of this example.

[10] `https://www.wbdg.org/resources/chp.php`

This is a Reinforcement Learning Problem [Sutton and Barto, 1998]. The variables a (resp. s) stand for 'action' (resp. 'state'). The policy p, given a state and a time, provides the next action. Given a state, an action and a time, the function $\mathfrak{T}$ provides the new state of the system, which depends of a random value ω. The total cost of the policy p over the T time steps is then COST_p. Note that we do not define formally the state and action spaces as the goal of the introduction is to give the main ideas and intuition behind the power systems challenges. A formal definition can be found in Chap. 8.

When $\omega_0, \ldots, \omega_{T-1}$ are Markovian random values, the problem above is termed *Markov Decision Process (MDP)*. Many techniques have been developed to tackle such problem [Bertsekas, 1995, Couëtoux, 2013]. The Markovian property of the random variables is a key point in some of these techniques named below. Note that when the process is non-Markovian, it can be made Markovian by enlarging the complexity of the random processes state. For example, if the random variable ω_t depends on the 10 last random variables $\omega_{t-1}, \ldots, \omega_{t-10}$, we define a new random variable: $\Omega_t = (\omega_t, \ldots, \omega_{t-9})$, and we rewrite the decision process with this new variable Ω: the process is Markovian.

[Saravanan et al., 2013] provides clear review on the optimization techniques used to solve the unit commitment problem. Among them, Stochastic Dynamic Programming (SDP) [Bellman, 1957], Stochastic Dual Dynamic Programming (SDDP) [Pereira and Pinto, 1991], Model Predictive Control (MPC) [Bertsekas, 2005] or Direct Policy Search (DPS) [Schoenauer and Ronald, 1994] are used to deal with MDP. However, each of them is limited:

- MPC is suboptimal by nature (deterministic approximation);
- SDP needs a moderate size of state space;
- SDDP requires convexity of Bellman values and a moderate complexity of random processes state (that is, either the random process is Markovian or it can be made Markovian as described above without increasing to much the random processes state).

SDP and SDDP generally use some linear optimization method (see Section 1.2.3). We would like to have a representation of the unit commitment problem closer from the reality and still have a method which reaches the optimal policy. That is, we need to relax the assumptions on the model. Namely, we want to get rid of the convexity conditions, to have an arbitrary large state space and an arbitrary random process.

Research Question #1

How can we handle the Unit Commitment Problem without assuming:

- the convexity or linearity of the cost function;
- the Markovianity (or equivalently, as discussed above, the moderate size complexity) of the random process?

Decision theory

The sub-equation

$$i^* = \arg\min_{i \in \mathscr{I}} \max_{u \in \mathscr{U}} \textsc{Cost}(i,u) \tag{1.3}$$

falls within decision theory.

If $\mathscr{I} = \{i_1, \dots, i_n\}$ (resp. $\mathscr{U} = \{u_1, \dots, u_m\}$) is the finite set of possible investments (resp. non-stochastic uncertainties), i.e. using the vocabulary of decision theory, the set of *policies* or *strategies* (resp. *scenarios*), then Eq. 1.3 is called *Wald criterion* [Liu, 2015]. As explained in Section 1.2.1, this is the worst case scenario. "Typically the worst case is a nuclear war and everybody is dead ... so that there is no point in optimizing anything"©[11]. More formally, the assumption behind the use of this criterion is that *the Nature*[12] knows in advance what will be our strategy. This is overall a very conservative criterion and the price for this robustness might be high [Bertsimas and Sim, 2004]. So it is worth taking a look at other tools for decision under uncertainties, such as the *Savage criterion* or *scenario-based planning*. We refer to [Liu, 2015] for a good introduction of these notions.

Instead of the Wald criterion (Eq. 1.3), investments against non-stochastic uncertainties can be modelled as an *adversarial zero-sum matrix game*. Given a $n \times m$ matrix $\left(M_{j,k}\right)_{j,k}$:

- we choose (privately) an investment strategy $i_j \in \{i_1, \dots, i_n\}$, i.e. a row $j \in \{1, \dots, n\}$;
- the Nature chooses a scenario $u_k \in \{u_1, \dots, u_m\}$ (i.e. a column $k \in \{1, \dots, m\}$) without observing i_j;

[11] Olivier Teytaud.

[12] In decision theory, the choice of u can be called *Nature's choice*.

- we receive a reward[13] $M_{j,k}$ and the Nature receives reward $1 - M_{j,k}$.

In this setting, interesting policies are often not deterministic. We play a stochastic policy $p \in [0,1]^n$ ($\sum_{j=1}^n p_j = 1$) and Nature plays a stochastic policy $q \in [0,1]^m$ ($\sum_{k=1}^K q_k = 1$). That is, we actually play j with probability p_j and k is (independently) played by Nature with probability q_k. Our expected payoff is therefore $p^t M q = \sum_{j,k} p_j M_{j,k} q_k$, and the expected payoff for Nature is $1 - p^t M q$. A *Nash equilibrium* is a pair (p,q) such that

$$\forall (p', q'),\ p'^t M q \leq p^t M q \leq p^t M q'.$$

Intuitively speaking, at a Nash equilibrium, neither Nature nor us can improve our expected payoff by changing our strategy. It is known, see [Nash, 1951], that with n and m finite,

- there is always at least one Nash equilibrium;
- it is not necessarily unique;
- all Nash equilibria (p,q) lead to the same value $v = p^t M q$.

A classical problem is thus the evaluation of a Nash equilibrium, or an approximation thereof in a reasonable time. We want furthermore include the stochasticity ω. In this setting, instead of getting a fix reward $M_{j,k}$, we will get a *stochastic reward* $M_{j,k}(\omega)$ (see formal definition in Chap. 9).

Research Question #2

How can we approximate in a reasonable computational time a Nash equilibrium in the stochastic adversarial case?

However, we point out that a resulting Nash strategy is in general a *mixed strategy*, i.e. a probability distribution over the strategies in $\mathscr{I}$ opposed to a *pure strategy* which is a mixed strategy with probability 1 over one element of $\mathscr{I}$. This is classical in games theory. However, in the context of power system investments, it seems difficult to prescribe "huge investment of offshore wind power with probability $1/3$", so that a reflection has to be carried out on the adaptivity of such criterion to the power system setting.

[13] we keep the vocabulary of game theory, so that the reward corresponds to $-$COST

Statistics and Noisy Black-Box Optimization

Last but not least, we need to handle the operator $\mathbb{E}$. As pointed out in Section 1.2.1, either we have access only to some finite archive, i.e. $(\omega_1, \ldots, \omega_n)$ realizations of the random variable ω of unknown probability distribution Π, or a generative model is available.

In the first case, optimizing against this finite sample leads to a *bias*. It is then interesting to estimate this bias, using methods coming from the statistical community such as Bootstrap (BS) [Efron, 1982], Jackknife (JK) [Quenouille, 1949], Cross-Validation (CV) [Arlot and Celisse, 2010], and then provide a better, corrected, expertise. This is explained in Chap 10.

Research Question #3

In the context of capacities expansion planning, how can we reduce the bias resulting from a finite (small) archive?

Regarding the latter case, this means that, given an investment i and a policy p, we can get a value $\textsc{Cost}(i, p, \omega)$, where ω is for example a weather realization that we do not know in advance. This kind of optimization is called *black-box noisy continuous optimization*, as we do not assume anything on the cost function. Black-box noisy continuous optimization is formalized in Chap. 2. We focus on black-box noisy optimization algorithms, comparing their optimal rates of convergence.

Research Question #4

What are the optimal convergence rates of various families of black-box noisy continuous optimization algorithms?

In particular, continuous black-box noisy optimization will be used in Chap 8, and it can be seen as a substitution of the classical cutting plane method. Another field of application of this method, related to power systems, is the Direct Policy Search method, i.e. the search for an optimal parameter x^* of a parametric function f_x, see [Kormushev and Caldwell, 2012]. This justifies the extensive interest for this optimization method in this thesis: we need a fast and robust black-box noisy optimization algorithm!

1.3 Structure of the thesis & contributions

Chap. 1 provides the motivations behind this Ph.D. thesis and the fields of application. Afterwards, this document is divided in three parts. Part I targets the continuous black-box noisy optimization problem and Part II concerns uncertainties, termed *noise*, with other kinds of features. Part III concludes and gives some perspectives. Appendix A summarize the main notations of the thesis. In the rest of this section, text in bold indicates the notions that will be defined in the following chapters.

1.3.1 Contribution in noisy AND black-box continuous optimization

Part I is dedicated to the study of continuous black-box noisy optimization algorithms. It handles the 4th research question:

Research Question #4

What are the optimal convergence rate of various families of black-box noisy continuous optimization algorithms?

Chap. 2 introduces the continuous black-box noisy optimization setting, that is, defines this problem, summarizes the state of the art and enlightens a few challenging questions of this field. Chap. 3 studies the rate of convergence of a given family of algorithms - **Evolution Strategies (ESs)** - in the case of small noise. Chap. 4 exhibits a lower bound for a large family of **comparison-based algorithms** in the **additive noise** setting. Chap. 5 discusses the possibility for a comparison-based algorithm to be as fast as a **value-based algorithm** in the black-box continuous noisy framework. Chap. 6 analyses the rates of convergence of a value-based algorithm, the **Newton-like algorithm**, when confronted with different kind of noises (small, moderate, big). Last, Chap. 7 proposes a method to select, among different noisy optimization algorithms, the one with an optimal rate of convergence for the optimization problem at hand.

1.3.2 Delicate cases

Part II handles arduous cases, when the optimization, still black-box and noisy, has additional constraints.

Contribution in noisy AND time-dependent optimization

Chap. 8, regards the 1st research question.

Research Question #1

How can we handle the Unit Commitment Problem without assuming:

- the convexity or linearity of the cost function;
- the Markovianity (or equivalently, as discussed above, the moderate size complexity) of random process?

In addition to be noisy and black-box, we aim at tackling long term effects in Model Predictive Control (MPC is deterministic, see Section 1.2.3). We study a Reinforcement Leaning method, called *Direct Model Predictive Control (DMPC)*. Assuming the convergence of the noisy optimization routine, it provably reaches an optimal policy without linearity or convex assumptions on the cost and transition functions, and without requiring moderate complexity of the random values state. We also compare the performances of DMPC and MPC on a multiple-battery management problem, and two hydroelectric problems.

Contribution in noisy AND adversarial optimization

We consider the 2nd research question in Chap. 9.

Research Question #2

How can we approximate in a reasonable computational time a Nash equilibrium in the stochastic adversarial case?

Chap. 9 analyses the adaptation of the **Grigoriadis & Khachiyan algorithm** [Grigoriadis and Khachiyan, 1995] to the stochastic case, in the setting where the complexity measure is the *number of evaluations*. We also introduce variants of this algorithm and a new algorithm, proving their theoretical validity and testing their experimental efficiency.

Contribution in noisy AND finite-sample optimization

Last, Chap. 10 treats the 3rd research question,

Research Question #3

In the context of capacities expansion planning, how can we reduce the bias resulting from a finite (small) archive?

The optimization of capacities in large scale power systems is a stochastic problem, because the need for storage and connections varies a lot from one week to another and from one winter to another. It is usually tackled through **Sample Average Approximation (SAA)**. However, in many cases, data is high-dimensional: the **sample complexity** increases linearly with the number of parameters and can be scarcely available at the relevant scale. This leads to an underestimation of capacities. We suggest the use of **bias correction** in capacity estimation.

I

Contributions in Continuous Black-Box Noisy Optimization

Chapter 2

Background review

Numerical optimization or simply optimization of a real-valued function f, termed *objective function*, is the research of a point, such that the value of f at this point, called *fitness value*, is optimal. Without loss of generality, we will assume that the optimum is a minimum[1], since minimizing f is equivalent to maximizing $-f$. Hence we are looking for the minimizer x^* - supposed to be unique - such that for all x in the search space $\mathscr{D} \subset \mathbb{R}^d$,

$$f(x) \geq f(x^*). \tag{2.1}$$

Among the classical challenges encountered by continuous optimization, we identify multi-modality, non differentiability or non continuity, ill-conditioning, non-separability, high dimensionality, noise and constraints [Chotard, 2015, Auger, 2016].

2.1 The different flavours of optimization

2.1.1 White, gray or black?

When optimizing an objective function, the optimizer can have access to different amounts of information. *The white-box optimization problem* is the best possible scenario: given a search point, one can get the fitness value at this point, as well as the gradient and the Hessian. More generally, we have access to the source

[1] unless specified otherwise.

code of the objective function. First (resp. second) order methods denote algorithms using the gradient (resp. the gradient and Hessian) of the objective function [Nesterov, 2004].

However, this ideal scenario is not often encountered in practice. More often, the practitioner has at best some partial information about the structure of the objective function, e.g. the smoothness or the separability property. This is *the gray-box optimization problem* [Whitley, 2015].

In a real world optimization problem, a common setting is to obtain only the fitness values of the objective function: this is *the black-box problem*. No knowledge about the internal process involved in the objective function can be exploited: given a point, an oracle returns the corresponding fitness value. The black-box setting is natural in many industrial applications, where the fitness value results of some heavy simulation or some executable file. Black-box algorithms belong to zero order methods, also known as derivative free optimization methods. Note that in the literature, fitness-value based algorithms approximating the gradient (resp. Hessian) by finite differences are considered either as first (resp. second) order methods or as zero order methods.

2.1.2 Global or local?

It is classical to make a dichotomy between global and local optimization. The objective of global optimization is to find the global optimum from any starting point whereas local optimization is the search of a minimum in the vicinity of a starting point. In particular, local optimization faces the risk to be stuck in local extrema. Evolutionary algorithms[2] are often labelled as global optimization algorithms in contrast to gradient-descent algorithms, assumed to be local optimization methods. The global convergence can be trivial to prove, e.g. grid search. However, what matters in our studies is precisely the *speed* of convergence, as we can not be satisfied with a slow one. In the present document, we consider smooth objective functions with a unique optimum - thus local convergence.

2.1.3 Noisy or noise-free?

The setting of Eq. 2.1 assumes that the oracle returns the exact value of f in x: this is the so called *noise-free setting*. However, the returned fitness value might be

[2] see Section 2.3 for an introduction to these algorithms.

perturbed or inaccurate, due to some measurement errors, to the sensor's sensitivity or to other stochastic effects such as random simulations in games or stochastic weather in power systems. This is termed *noise*, and the optimization problem is then a noisy optimization problem. Formally, the noisy objective function can be modelled by a stochastic process. Given a search point $x \in \mathscr{D}$, the oracle provides the fitness value $f(x, \omega)$, where ω is a random variable independently sampled at each call to the black-box. In this setting, the optimization in the manner of Eq. 2.1 does not make any sense. The optimization of a noisy objective function is the search for the minimizer x^* such that for all $x \in \mathscr{D}$,

$$\mathbb{E}_\omega f(x, \omega) \geq \mathbb{E}_\omega f(x^*, \omega), \tag{2.2}$$

where $\mathbb{E}_\omega$ denotes the expectation operator over ω.

Note that what we term *noisy optimization* in the present document is sometimes called *stochastic optimization* in the literature. This is misleading since stochastic optimization also refers to optimization algorithms relying on internal stochastic processes, such as evolutionary algorithms.

Models of noise

In the presentation of the different noise models below, we denote by ω a random variable, sampled independently with a given probability distribution at each new evaluation of a search point.

Actuator noise. When the search point is corrupted by noise, this is termed *actuator noise*. The noisy objective function is then:

$$f(x, \omega) = f(x + \omega) \tag{2.3}$$

The study of the actuator noise is beyond the scope of this thesis, however an extended analysis can be found in [Jin and Branke, 2005, Beyer, 2004].

Additive noise. The additive model of noise is by far the most natural and studied noise model. To the best of our knowledge, it has been formulated for the first time in [Hotelling, 1941], motivated by some practical applications in agriculture, industry and economy. It is formalized in the following way:

$$f(x, \omega) = f(x) + \omega. \tag{2.4}$$

In this model, the noise has lower bounded variance, even in the neighbourhood of the optimum. In most cases, ω is a standard normal variable, and this is termed *Gaussian noise* [Arnold and Beyer, 2002, Astete-Morales et al., 2014, Arnold and Beyer, 2001]. Cauchy and χ^2 distributions have also been studied [Arnold and Beyer, 2006, Arnold and Beyer, 2003]. They conclude that there is no significant differences between a Gaussian and Cauchy noise, but the dynamics of the χ^2 is distinct, due to the asymmetry of the distribution. Other studies simply assume that the variance of ω is bounded by a constant [Fabian, 1967, Shamir, 2013] - this is called heavy tail noise.

In discrete noisy optimization[3], [Dang and Lehre, 2015] found similar behaviour of Evolutionary Algorithm, independently of the Gaussian, uniform or exponential distribution of the additive noise. [Akimoto et al., 2015] studied discrete noisy objective functions with Gaussian noise and heavy tail noise.

Multiplicative noise. The multiplicative noise has been studied in [Arnold and Beyer, 2002, Jebalia et al., 2011]. It refers to:

$$f(x,\omega) = f(x)(1+\omega). \tag{2.5}$$

If the probability distribution of ω is conveniently lower bounded, then some standard $(1+1)$-Evolution Strategy[4] converges to the optimum. If arbitrary negative values can be sampled with non-zero probability, then it does not converge [Jebalia et al., 2011].

Bernoulli setting. Another branch of noisy optimization (here maximization) involves Bernoulli variable as objective functions: for a given search point x,

$$f(x,\omega) = \begin{cases} 1 \text{ with probability } \mathbb{E}_\omega f(x,\omega), \\ 0 \text{ otherwise.} \end{cases} \tag{2.6}$$

For example, if $f(x,\omega)$ is a Bernoulli variable of parameter $\|x-x^*\|$, then $f(x,\omega) = 1$ with probability $\|x-x^*\|$.

Optimizing is then finding x such that $\mathbb{E}_\omega f(x,\omega)$ is maximum. This framework is particularly relevant in games [Coulom, 2011, Chaslot et al., 2008]. If x is a parameter of a game strategy, playing one match with this parametrization will result in a win (1), or a loss (0) and the random variable ω states

[3]Discrete optimization is the setting in which the objective function f has discrete variables, usually, $f : \{0,1\}^d \to \mathbb{R}$ or $f : \{0,1,\ldots,n\}^d \to \mathbb{R}$.

[4]see the definition in Section 2.3.2.

for the stochasticity in the game (e.g. in case of randomized policy). We then aim to find x with maximizes the probability of win. Problems tackled by Direct Policy Search, such as viability problems or binary control problems [Aubin, 2009, Chapel and Deffuant, 2006], involve this kind of optimization. In particular, if the optimal policy has a success rate of 100%, then the variance decreases to zero in the neighbourhood of the optimum.

Generalization of additive and multiplicative models of noise. A more general model of noise can be formalized in the following way:

$$f(x,\omega) = f(x) + (f(x) - f(x^*))^z \omega, \tag{2.7}$$

When $z = 0$ (resp. $z = 1$ and $f(x^*) = 0$), we get the additive (resp. multiplicative) noise. When $z > 0$, the noise decreases to zero near the optimum. This setting is not artificial as we can observe this behaviour in many real problems, as explained for the Bernoulli noise. Three chapters are devoted to the study of additive noise, one chapter focuses on small noise - multiplicative and more ($z \geq 1$), and one chapter encompasses several kinds of noise ($z \in 0, 1, 2$).

Adaptation to the noise

Three main features emerge to cope with noise: adjusting the *population* parameters vs. *averaging* the search points vs. using a *surrogate model*.

A classical scheme of optimization algorithms is to generate a population of search points from a central point at each iteration. We can increase this population size to tackle the effect of noise. It is also possible to increase its *variance* or mutation strength. This latter technique prevents premature convergence. Importantly, the population can escape some sub-optimal search regions. We refer to [Hansen et al., 2009, Arnold and Beyer, 2001, Arnold, 2002, Arnold and Beyer, 2000b, Arnold and Beyer, 2000a, Fitzpatrick and Grefenstette, 1988, Arnold and Beyer, 2006] for more details.

Resampling means that the query to the black-box is repeated several times for a given search point [Aizawa and Wah, 1993, Aizawa and Wah, 1994, Beyer, 1993, Hammel and Bäck, 1994]. Afterwards, some statistic of the repeated sample is used as the approximate fitness function of the point. In general, it is the average. For a given point $x \in \mathscr{D}$ and an integer r, the approximate fitness value y is:

$$y = \frac{1}{r}\sum_{i=1}^{r} f(x, \omega_i),$$

where $(\omega_i)_{1\leq i\leq r}$ are r i.i.d realizations of the random variable ω. Averaging decreases the variance - denoted Var - of the fitness value. Therefore, if $Var(f(x,\omega)) = \sigma^2 > 0$, then $Var(y) = \frac{\sigma^2}{r}$. Resampling is then a way to reduce the uncertainty of the fitness value. The key point is how to choose optimally r, the number of resamplings, as increasing the number of calls to the black-box increases the computational burden. This number can be fixed, or can increase exponentially with the iteration index [Astete-Morales et al., 2014], or can be adaptive [Astete-Morales et al., 2014, Branke and Schmidt, 2003, Branke and Schmidt, 2004, Cantú-Paz, 2004]. Resampling can be viewed as an averaging over time. An alternative method, which can be seen as an averaging over space, consists in evaluating the fitness by averaging over the neighbourhood of the search point [Fabian, 1967, Jin and Branke, 2005]. This is based on the assumptions that the objective function is smooth and that the noise distribution is the same at least in the neighbourhood of a point.

Contradictory results are found in the literature regarding the problem to choose between increasing the population size or resampling. So far, it seems that the relevance of one method or the other is problem-dependent [Jin and Branke, 2005, Chotard, 2015]. Merging the two methods is also a solution [Miller, 1997, Miller and Goldberg, 1996].

A third trend is to build a model - called surrogate model [Ong et al., 2003, Zhou et al., 2004] - of the noisy objective function by using the previous search points. [Branke et al., 2001] uses local regression. Confident Local Optimization (CLOP) [Coulom, 2011] also performs local regression. In practice, it is robust to high noise and does not require specific parameter tuning. However, we are not aware of a mathematical analysis. NEWUOA has been developed by Powell [Powell, 2004, Powell, 2008]. It performs some quadratic interpolation in order to draw a model of the objective function. It is very efficient in the noise-free case, but slower in the noisy setting (see [Moré and Wild, 2009]). QLR - for Quadratic Logistic Regression - is based on a Bayesian quadratic local regression. In particular, it can sample points far from the current recommendation, which is a crucial point. It is designed for Bernoulli noise, thus very efficient on such a setting - it can then be considered as a gray-box optimization method. We refer to [Chaloner, 1989, Fackle Fornius, 2008, Khuri et al., 2006,

Schein and Ungar, 2007] for more on this algorithm.

2.2 Convergence(s): what makes an algorithm 'good'?

An optimization algorithm must provide a 'good' approximation of the optimum. What does it mean? Contrarily to discrete optimization, a continuous optimization algorithm does not in general reach the optimum. It outputs successive estimations of the optimum, which should converge toward to optimum in a 'reasonable' time, i.e. as 'fast' as possible. But being fast or not depends on which measure we consider to be important. Regarding some industrial applications, a call to the black-box might be expensive, requiring heavy computations. Thus, the goal is to find a good approximation of the optimum within a number of calls as small as possible. That is why from now on, unless specified otherwise, the variable indexation in the document is always in the number of evaluations, i.e. calls to the oracle.

2.2.1 Exploration vs. exploitation

In the noisy black-box scenario, an optimization algorithm generates several sequences:

- $x_1, x_2, \ldots, x_n, \ldots$, the successive search points, or evaluation points;
- $y_1, y_2, \ldots, y_n, \ldots$, their corresponding noisy fitness values: $y_m = f(x_m, \omega)$;
- $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n, \ldots$, the successive *recommendations* or *approximations* of the optimum x^*, where $\tilde{x}_m$ is provided after m fitness evaluations have been performed.

Each search point x_m is the output of a computable function of the previous search points and their respective function values. The computation of the search point may involve random processes if the algorithm is randomized or the objective function is stochastic. Even though in most cases, the recommendation and the search points are exactly the same, it is crucial to distinguish between these two types of points. In particular, in the noisy setting, ignoring this difference can lead to poor results [Fabian, 1967, Coulom, 2011]. The sequence $(x_1, x_2, \ldots, x_n, \ldots)$ represents the exploration phase and the sequence

$(\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n, \ldots)$ corresponds to the exploitation one. A key point is how to allocate resources (i.e. number of calls to the black-box in our setting) to each of them - especially how to guarantee an efficient exploration - and how to choose these sequences.

2.2.2 Convergence criteria

We review some general criteria, which can be applied to any optimization algorithm. Note that measurements specific to one type of algorithm, such as progress rate for Evolution Strategy (ES) (see [Beyer, 2001]), are not discussed here.

Uniform Rate (UR)

Looking at the distance to the optimum is the first criterion that comes to mind. It is called *Uniform Rate*. The term *rms* - for *root mean square* - is also encountered in the literature.

Definition 2.2.1 (Uniform Rate (UR))**.** *Using the previous notations, the Uniform Rate is defined by:*

$$UR_n := \|x_n - x^*\|. \tag{2.8}$$

In particular, we consider the search points. As a consequence, having a 'good' Uniform Rate implies to sample the search points only close to the optimum. Additionally, due to some randomization of the optimization algorithm, the Uniform Rate can be a random variable.

Regrets

The concept of *Regret* is widely used in the bandit literature. It is also used in the optimization framework, sometimes under other names or without specific name. Basically, the regret accounts for the 'loss' or 'cost' of choosing the search or approximation point instead of the optimum. Therefore, we measure the difference between the point used or recommended by the algorithm and the optimum in terms of objective function.

The most usual form of regret is termed *Simple Regret (SR)*. It is widely used, possibly without this name [Bubeck et al., 2009]. It focuses only on approximating, with recommendations, the optimum in terms of fitness values.

Definition 2.2.2 (Simple Regret (SR))**.** *Using the previous notations, the Simple Regret is defined by:*

$$SR_n := \mathbb{E}_\omega(f(\tilde{x}_n, \omega) - f(x^*, \omega)) = f(\tilde{x}_n) - f(x^*). \tag{2.9}$$

The expectation operates only on the noise ω in $f(\tilde{x}_n, \omega)$, and not on $\tilde{x}_n$. As a consequence, SR_n is a random variable due to the stochasticity of the noisy evaluations of the search points or the possible internal randomization of the optimization algorithm. In the noise-free case, it can be used to determine the precision of a method, by ensuring that the algorithm outputs a recommendation $\tilde{x}_m$ satisfying $SR_m \leq \varepsilon$, for a given $\varepsilon > 0$.

Some benchmarks, notably the Bbob/Coco framework in the first version, did not allow the distinction between search points and recommendations, so that the Simple Regret can not be checked. An alternative definition, that aims to measure the precision in a similar way to *SR*, is the *Approximate Simple Regret*.

Definition 2.2.3 (Approximate Simple Regret (ASR))**.** *Using the previous notations, the Approximate Simple Regret is defined by:*

$$ASR_n := \min_{m \leq n} f(x_n) - f(x^*). \tag{2.10}$$

It is used in the Bbob/Coco framework [Auger et al., 2010a, Auger et al., 2010b, Auger et al., 2010c, Finck and Beyer, 2010, Ros, 2010a, Ros, 2010b, LaTorre et al., 2010, Tran and Jin, 2010, Hansen and Ros, 2010], and in some theoretical papers [Dang and Lehre, 2015]. *ASR* takes into account the 'best' evaluations among all the search points.

Another form of regret is the *Cumulative Regret (CR)*. This criterion keeps track of the loss of every search point, not only the best.

Definition 2.2.4 (Cumulative Regret (CR))**.** *Using the previous notations, the Cumulative Regret is defined by:*

$$CR_n := \sum_{i=1}^{n} \left(f(x_i) - f(x^*) \right). \tag{2.11}$$

The Cumulative Regret is relevant for, e.g., online optimization of a factory, online optimization of medical treatments, and all cases in which each function evaluation is an actual loss and not only a simulated loss.

Expected running and hitting time

When testing algorithms, it is common to use some hitting time, i.e. to consider the first time that the sequence of points 'hits' a given subset of the search space. We distinguish the *first hitting time* and the *running time*.

Definition 2.2.5 (First hitting time)**.** *For a precision $\varepsilon > 0$, the first hitting time τ_ε is a random variable defined by:*

$$\tau_\varepsilon := \min\{n \in \mathbb{N} | \ \|\tilde{x}_n - x^*\| \leq \varepsilon\} \tag{2.12}$$

Definition 2.2.6 (Running time)**.** *For a precision $\varepsilon > 0$, the running time ρ_ε is a random variable defined by:*

$$\rho_\varepsilon := \min\{n \in \mathbb{N} | \ \max_{m \geq n} f(\tilde{x}_m) - f(x^*) \leq \varepsilon\} \tag{2.13}$$

The running time refers to the first 'stable' hitting time, i.e. the next recommendation is at least as good as the previous one. In the noise-free setting, if the recommendation is defined as $\tilde{x}_n = x_i(n)$ with $i(n) = \arg\min_{1 \leq i \leq n} SR_i$, then this is equivalent to finding the smallest integer n such that $SR_n \leq \varepsilon$. It is reasonable to assume that the recommendation is the optimal search point in the noise-free setting, as maintaining a best search point so far is easy and cheap. Unfortunately, in the case of noisy optimization, there is no such equivalence and there is no natural extension of running time without checking the infinitely many values SR_m for $m \geq n$.

[Corus et al., 2014, He and Yao, 2003, Akimoto et al., 2015] are devoted to the study of the expected hitting time and the expected running time.

2.2.3 Type of convergence

Regarding the regrets and the Uniform Rate, we distinguish two typical rates of convergence. The *log-log convergence* and the *log-linear convergence*, for which lower and upper bounds will be discussed in this thesis. In the following, when we do not specify the mode of convergence (a.s. or in expectation), then the definition or property holds for every mode.

Definition 2.2.7 (log-log convergence)**.** *The sequence of random variables $(R_n)_{n \in \mathbb{N}}$ is said to converge log-logarithmically (or logarithmically):*

- *almost surely if $\lim_{n \to +\infty} \frac{\log(R_n)}{\log(n)}$ exists almost surely and is negative.*

- *in expectation if* $\lim_{n\to+\infty} \frac{\log(\mathbb{E}(R_n))}{\log(n)}$ *exists and is negative.*

If the limits above exist but are positive, then we say that the algorithm diverges log-logarithmically.

Terminology & Notations. When $\limsup_{n\in\mathbb{N}} \frac{\log(R_n)}{\log(n)} = -\alpha$, with $\alpha > 0$, it is equivalent to write $R_n = O\left(\frac{1}{n^\alpha}\right)$ or to say that R_n is in $O\left(\frac{1}{n^\alpha}\right)$. If $\liminf_{n\in\mathbb{N}} \frac{\log(R_n)}{\log(n)} = -\alpha$, with $\alpha > 0$, it is equivalent to write $R_n = \Omega\left(\frac{1}{n^\alpha}\right)$ or to say that R_n is in $\Omega\left(\frac{1}{n^\alpha}\right)$. We called this number α the *slope* of R_n, denoted $s(R)$, and use this different notations throughout the present document. We discuss the optimal value of this slope α for *UR*, *SR* and *CR*, depending on the characteristic of the optimization algorithm and on the objective function. The log-log convergence is typical in the noisy optimization setting with lower bounded variance [Arnold and Beyer, 2002, Astete-Morales et al., 2014, Chen, 1988, Fabian, 1967, Coulom, 2011, Shamir, 2013, Decock and Teytaud, 2013].

Asymptotic & non-asymptotic regime. We distinguish two different regimes. The asymptotic one holds when

$$\exists C > 0,\ \exists\, n_0 \text{ such that } \forall\, n \geq n_0,\ R_n \ \& \ \frac{C}{n^\alpha},$$

where & stands for $\leq$ or $\geq$ and this inequality can be in expectation or a.s. The non-asymptotic regime holds when

$$\exists C > 0,\ \forall\, n \in \mathbb{N},\ R_n \ \& \ \frac{C}{n^\alpha}.$$

In Definition 2.2.7, we adopted the asymptotic setting. However, non-asymptotic results will be discussed in the review of literature below (Section 2.3), in Chapters 4 and 5 and in Appendix B. By default, rates of convergence are asymptotic.

Definition 2.2.8 (log-linear convergence)**.** *The sequence of random variables* $(R_n)_{n\in\mathbb{N}}$ *is said to converge log-linearly (or linearly):*

- *almost surely if* $\lim_{n\to+\infty} \frac{\log(R_n)}{n}$ *exists almost surely and is negative.*

- *in expectation if* $\lim_{n\to+\infty} \frac{\log(\mathbb{E}(R_n))}{n}$ *exists and is negative.*

When the limits above exist but are positive, then we say that the algorithm diverges log-linearly. The log-linear convergence is sometimes called *exponential* convergence since the sequence $(R_n)_n$ decreases exponentially fast in the number of calls to the oracle. Log-linear convergence is classical in noise-free zero-order optimization [Auger, 2005, Beyer, 2001, Teytaud et al., 2005]. [Hansen et al., 2015] has shown a link between the expected hitting time and the rate of convergence in case of log-linear convergence: $\mathbb{E}\tau_\varepsilon \sim \log(1/\varepsilon)/\alpha$, where $\alpha > 0$, with $\lim_{n\to+\infty} \frac{\log(R_n)}{n} = -\alpha$.

The log-linear convergence is faster than the log-logarithmic one, however, we can encounter even faster rates of convergence.

Definition 2.2.9 (Super-linear convergence)**.** *We say that an algorithm converges super-linearly of order α when:*

$$\lim_{n\to+\infty} UR_n = 0$$
$$\textit{and} \quad \lim_{n\to+\infty} \frac{UR_n}{UR_{n-1}^{\alpha}} = \mu > 0.$$

In particular, when $\lim_{n\to+\infty} \frac{\log(UR_n)}{n} = -\infty$, the super-linear convergence should be investigated. It is classical in the noise-free case with surrogate model [Auger et al., 2005]. If $\alpha = 2$, the convergence is quadratic. In the noise-free setting, Newton's algorithm is quadratic [Nesterov, 2004] and quasi-Newton methods such as BFGS are superlinear [Nesterov, 2004].

2.2.4 Discussion

***UR* and *SR*.** By definition, a good slope for the Uniform Rate ($s(UR)$) is harder to reach than for the *SR* because all search points must verify the bound, not only the recommended ones. For any problem, if for some algorithms, $s(UR) \leq c$, then for the same problem there is an algorithm such that $s(SR) \leq c$. A slope 0 for *UR* and *SR* can be trivially reached by an algorithm with constant $(x_n, \tilde{x}_n)$.

***CR* and *SR*.** An algorithm with constant $(x_n, \tilde{x}_n)$ will similarly provide a trivial slope 1 for *CR*. On the other hand, optimality between $s(SR)$ and $s(CR)$ can not be reached simultaneously. In discrete settings, this so called *trade-off* is proved by [Bubeck et al., 2011]. They show that in the framework of stochastic multi-armed bandit problems, the smaller the *CR*, the larger the *SR*. This trade-off is

also observed in continuous optimization algorithms, see Section 2.3.1. Using the vocabulary of the multi-armed bandit community, this is the trade-off between exploration and exploitation.

***UR* and *CR*.** We are not aware of differences between algorithms specialized on optimizing $s(UR)$ criterion and $s(CR)$ criterion.

2.3 Algorithms and state of the art

We introduce in this section some zero-order noisy optimization algorithms and discuss their performance regarding the criteria of Section 2.2.2. This review is not exhaustive as it is centred on the algorithms studied later in this thesis. [Chotard, 2015] provides a more complete overview, emphasizing the opposition between deterministic and stochastic algorithms. Stochastic algorithms generate recommendations through the use of random processes. This makes them more likely to escape local minima and provides them some sort of robustness - that will be discussed - when confronted with noise. In contrast, we are particularly interested in the dichotomy between comparison-based algorithms and algorithms using an approximate gradient or Hessian: *value-based algorithms*.

2.3.1 Value-based algorithms

Kiefer-Wolfowitz method

[Kiefer and Wolfowitz, 1952] have made a pioneering work to find numerically the optimum of a unidimensional noisy objective function such that $\mathbb{E}_\omega(f(x,\omega)) = f(x)$ with $Var(f(x,\omega))$ bounded. In particular, this covers the additive case of noise, but it is more general. The optimum was initially a maximum, but we write below the results in minimization. Their method derives from the work of [Robbins and Monro, 1951] one year before, devoted to solve the equation $\mathbb{E}_\omega f(x,\omega) = \alpha$. The Kiefer-Wolfowitz algorithm is a gradient descent algorithm, where the gradient is approximated by finite differences. Given two sequences of positive numbers $(a_n)_{n\geq 1}$ and $(c_n)_{n\geq 1}$, and an initial recommendation $\tilde{x}_0 \in \mathbb{R}$, the successive recommendations $\tilde{x}_n$ follow the **Kiefer-Wolfowitz scheme**.

$$\textbf{Kiefer-Wolfowitz scheme} \left| \begin{array}{l} \tilde{x}_{n+1} = \tilde{x}_n - a_n g_n, \\ g_n := \text{approximate gradient, estimated by finite differences.} \end{array} \right. \tag{2.14}$$

In the original method, [Kiefer and Wolfowitz, 1952] proposes to estimate g_n as follows:

$$g_n = \frac{f(\tilde{x}_n + c_n, \omega) - f(\tilde{x}_n - c_n, \omega)}{c_n}.$$

Note that in this part (Section 2.3.1), the recommendation $\tilde{x}_n$ is indexed in the number of *iterations*, and not in the number of calls to the black-box, as formalized in Section 2.2.1. This does not change the results on the rate of convergence, because there is a fixed number of calls to the oracle. It just adds a multiplicative constant in the upper bound, which does not modify the asymptotic slope.

Assuming that c_n decreases to 0 and that:

$$\sum_{n=1}^{+\infty} a_n = +\infty, \quad \sum_{n=1}^{+\infty} a_n c_n < +\infty \quad \text{and} \quad \sum_{n=1}^{+\infty} a_n^2 c_n^{-2} < +\infty; \tag{2.15}$$

under some strong regularity conditions of the objective (unidimensional) function, Kiefer and Wolfowitz proved the convergence in probability of the sequence $(\tilde{x}_n)_{n\geq 1}$ toward the optimum x^*, without a specific rate of convergence though. [Blum, 1954a, Blum, 1954b, Burkholder, 1956, Dvoretzky, 1956] successively broadened this convergence result to the multidimensional case and weakened the regularity conditions. The first optimal convergence rate was shown in [Dupač, 1958], using some inequalities from [Chung, 1954]. The expected Simple Regret has slope $O(n^{-1/2})$ when the objective function is twice differentiable and $O(n^{-2/3})$ when it is three times differentiable. However, this rate is not improved for functions with higher derivatives. It is formalized in Theorem 2.3.1.

Theorem 2.3.1 ([Dupač, 1957], Simple Regret of Kiefer-Wolfowitz algorithm)**.** *The (unidimensional) objective function has a unique minimizer x^* and satisfies:*

- $\forall \varepsilon > 0$, $\exists$ $K > 0$, $K' > 0$ *such that* $\forall$ $x \in B(x^*, \varepsilon)$, $K|x - x^*| \leq |f'(x)| \leq K'|x - x^*|$*;*
- $f''(x^*) > 0$ *exists.*

If the noise satisfies $\mathbb{E}_\omega(f(x,\omega)) = f(x)$ and $Var(f(x,\omega))$ is bounded, if $a_n = \frac{a}{n}$ and $c_n = \frac{c}{n^{1/4}}$ with $a > 0$ big enough and $c > 0$ small enough, then

$$\mathbb{E}(\tilde{x}_n - x^*)^2 = O\left(\frac{1}{\sqrt{n}}\right) \tag{2.16}$$

and this is the optimal rate. Additionally, if f''' exists and is bounded in the neighbourhood of x^*, then with a_n as previously and $c_n = \frac{c}{n^{1/6}}$,

$$\mathbb{E}(\tilde{x}_n - x^*)^2 = O\left(\frac{1}{n^{2/3}}\right) \tag{2.17}$$

and this rate is optimal. In particular, when f is smooth enough, we get $s(SR) = -2/3$.

Note also that [Dupač, 1957] has shown that if the objective function is analytic and symmetric in the neighbourhood of x^*, then for any arbitrary $\varepsilon > 0$, $\mathbb{E}(\tilde{x}_n - x^*)^2 = O\left(\frac{1}{n^{1-\varepsilon}}\right)$. We refer to [Schmetterer, 1961] for a good survey of the early Kiefer-Wolfowitz-like algorithms.

Fabian's algorithm.

[Fabian, 1967] uses the same pattern as Kiefer and Wolfowitz, but computes the approximate gradient using averaging over space, which improves the rate of convergence. Fabian's algorithm follows the **Kiefer-Wolfowitz scheme** as in Eq. 2.14, and given an even integer s, the gradient $g_n = (g_n^{(i)})_{1\leq i\leq d}$ is updated as follows:

$$\forall\, i \in \{1,\dots,d\},\ g_n^{(i)} = \frac{1}{c_n}\sum_{k=1}^{s/2} v_k\left(f(\tilde{x}_n + c_n u_k e_i, \omega) - f(\tilde{x}_n - c_n u_k e_i, \omega)\right), \tag{2.18}$$

where $(e_i)_{1\leq i\leq d}$ is the standard basis and $(u_k)_{1\leq k\leq s/2}$ are such that $0 < u_1 < \cdots < u_{s/2} \leq 1$. $(v_k)_{1\leq k\leq s/2}$ are some weights, see [Fabian, 1967] for their computation. It has been shown in [Fabian, 1967] that this algorithm has SR arbitrarily close to $O(1/n)$ in expectation and also a.s., as formalized in Theorem 2.3.2.

Theorem 2.3.2 ([Fabian, 1967], Simple Regret of Fabian's algorithm)**.** *Let s be an even positive integer and $\varepsilon > 0$. Let x^* be the unique minimizer of the objective function f. The objective function f satisfies the following properties:*

- *its* $(s+1)^{th}$ *derivative exists on* $B(x^*, 2\varepsilon)$ *and is bounded there;*
- *its Hessian exists, is bounded in norm on* $\mathbb{R}^d$ *and it is positive definite and continuous at* x^**;*
- *its first derivative is zero at* x^**;*
- $\forall\ \varepsilon > 0$*, there exist* $\rho(\varepsilon) > 0$ *such that* $f(x) - f(x^*) \geq \rho(\varepsilon)$ *and* $\|Df(x)\| \geq \rho(\varepsilon)\ \forall x \in \mathbb{R}^d \setminus B(x^*, \varepsilon)$.

Assume that the noise satisfies $\mathbb{E}_\omega(f(x,\omega)) = f(x)$ *and that* $Var(f(x,\omega))$ *is bounded. Assume that* $a_n = \frac{a}{n}$ *and* $c_n = \frac{c}{n^\gamma}$*, with* $a > 0$*,* $c > 0$ *and* $0 < \gamma < 1/2$*. Assume that* $2\lambda_0 a > \beta_0$ *where* λ_0 *is the smallest eigenvalue of the Hessian and* $\beta_0 = \min(2s\gamma, 1-2\gamma)$*. Then, with* $\tilde{x}_n$ *obtained by* **Kiefer-Wolfowitz scheme** *(Eq. 2.14) with Eq. 2.18, a.s.:*

$$\lim_{n\to+\infty} n^\beta(\tilde{x}_n - x^*) = 0\ \forall\ \beta < \beta_0/2 \tag{2.19}$$

In particular, when f *is smooth enough, we get* $s(SR) = -2\beta$.

Note that SR is optimal when $\gamma = \frac{1}{2}(s+1)^{-1}$. In this case, $\beta_0 = \frac{s}{s+1} \underset{s\to\infty}{\to} 1$: β_0 can be made arbitrarily close to 1, so 2β also, but then γ goes to 0.

Depending on the value of γ, we get a good SR or a good CR, but never both simultaneously. In the case of quadratic functions with additive noise

- $\gamma \to \frac{1}{4}$ leads to $SR = O(n^{-1/2})$ and $CR = O(n^{1/2})$ a.s. and in expectation;
- $\gamma \to 0$ leads to $SR = O(n^{-1})$ and $CR = O(n)$ a.s. and in expectation.

We incidentally find out the trade-off discussed above.

On the computational side, the number of calls to the oracle per iteration is $2 \times s \times d$, which might become intractable in practice for large dimension and when s goes to infinity.

Spall's algorithm

[Spall, 1987] developed a method, called SPSA for *Simultaneous Perturbation Stochastic Approximation*, which alleviates by far this burden. It requires only 2 calls to the black-box at each iteration. Spall's algorithm follows the **Kiefer-Wolfowitz scheme** to update the recommendation as in Eq. 2.14 and the approximate gradient $g_n = (g_n^{(i)})_{1\leq i\leq d}$ is estimated in the following way:

$$\forall\, i \in \{1,\dots,d\},\ g_n^{(i)} = \frac{f(\tilde{x}_n + c_n\Delta, \omega) - f(\tilde{x}_n - c_n\Delta, \omega)}{2c_n\Delta^{(i)}} \tag{2.20}$$

where $\Delta \in \mathbb{R}^d$ is a vector of d mutually independent random variables. The $(\Delta^{(i)})_{1\le i\le d}$ have mean zero and satisfy others assumptions; in particular, first and second inverse moments must be bounded (see [Spall, 1992]). Incidentally, it excludes uniform and normal random variables. Usually, we consider Bernoulli random variables, see more in [Kleinman et al., 1999, Spall, 2000]. In terms of calls to the black-box, variants of Spall's algorithms have the same optimal convergence rate as Kiefer-Wolfowitz algorithm, i.e. $O(n^{-2/3})$ for the expected *SR*; it is formalized in Theorem 2.3.3.

Theorem 2.3.3 ([Gerencsér, 1999], Simple Regret of Spall's algorithm)**.** *Let $\beta = \min(4\gamma, 1-2\gamma) > 0$, $a_n = \frac{a}{n}$ and $c_n = \frac{c}{n^\gamma}$ with $a > 0$ and $c > 0$. Assume that the smallest eigenvalue of the Hessian matrix of the objective function f at x^*, denoted by α satisfies $a\alpha > \beta/2$. The objective function satisfies the following conditions:*

- *it has a unique minimizer x^* and is 3 times continuously differentiable in the neighbourhood of x^*;*
- *its gradient is defined in the neighbourhood of x^* and has continuous partial derivative up to second order*[5]*.*

Assume that the noise is additive: $\forall x \in \mathscr{D},\ f(x,\omega) = f(x) + \omega$, with ω a bounded random variable. Assume that the components of Δ are i.i.d, symmetrically distributed, bounded and the inverses of its higher moment are bounded. Then, with $\tilde{x}_n$ obtained by the **Kiefer-Wolfowitz scheme** *(Eq. 2.14) with Eq. 2.20, it follows:*

$$\mathbb{E}\|\tilde{x}_n - x^*\|^2 = O\left(\frac{1}{n^\beta}\right). \tag{2.21}$$

This rate is optimal for $\gamma = 1/6$, which give $\beta = 2/3$. In particular, when f is smooth enough, we get $s(SR) = -2/3$.

It was shown in [Spall and Cristion, 1998] that this rate is tight.

[5]The whole assumption involves some condition on the solution of a differential equation implying the gradient, we refer to [Gerencsér, 1999] for the details.

Spall also proposed some algorithms using additionally some approximate Hessian, i.e. approximated by finite differences, without improvement of the rate of convergence [Spall, 2000].

Spall's algorithms are also efficient in the noise-free setting, providing non-trivial rates of convergence.

Polyak-Tsybakov's algorithm

[Polyak and Tsybakov, 1990] proposed a method to merge the good properties of Fabian and Spall's algorithms - that is a *SR* in $O(n^{-\frac{s}{s+1}})$ with only 1 or 2 evaluations per iteration. They update the recommendation following the **Kiefer-Wolfowitz scheme** (Eq. 2.14) and estimate the gradient through the use of a kernel:

$$g_n = K(\Delta)\frac{f(\tilde{x}_n + c_n\Delta, \omega) - f(\tilde{x}_n - c_n\Delta, \omega)}{2c_n}. \tag{2.22}$$

K is a differential kernel. In practice, it is determined by using Legendre polynomials and Δ is a random vector uniformly distributed in $[-1/2, 1/2]^d$. With this method, the algorithm reaches an expected *SR* in $O(n^{-1})$ asymptotically for a wide family of functions, see Theorem 2.3.4.

Theorem 2.3.4 ([Polyak and Tsybakov, 1990], Simple Regret of Polyak-Tsybakov's algorithm)**.** *Assume that the objective function has a unique optimum at $x^*(f)$ and satisfies:*

1. *f has continuous partial derivatives up to order s inclusive, which satisfy the Hölder condition of order $\alpha \in (0,1]$;*
2. *$\forall\, x \in \mathbb{R}^d$, $(Df(x), x - x^*) \geq A_1\|x - x^*\|^2$;*
3. *$\forall\, x, x' \in \mathbb{R}^d$, $\|Df(x) - Df(x')\| \leq A_2\|x - x'\|$,*

where A_1 and A_2 are finite positive constants, $A_2 > A_1$. Let $\mathscr{F}$ denote the family of functions with a unique optimum satisfying these 3 conditions. Let $\beta = s + \alpha \geq 2$, $a_n = \frac{a}{n}$ and $c_n = \frac{c}{n^{1/2\beta}}$, $c > 0$, $a > (\beta - 1)/2A_1\beta$. Assume that the noise is additive, i.e. $\mathbb{E}(f(x,\omega) = f(x))$, with $\mathbb{E}(\omega) = 0$ and $\mathbb{E}(\omega^2)$ bounded. Assume that $\tilde{x}_n$ is obtained by the **Kiefer-Wolfowitz scheme** *(Eq. 2.14) with Eq. 2.22, it follows:*

$$\sup_n \sup_{f \in \mathscr{F}} n^{(\beta-1)/\beta}\mathbb{E}\|\tilde{x}_n - x^*(f)\|^2 < \infty \tag{2.23}$$

In particular, when f is smooth enough, we get $s(SR) = -(\beta - 1)/\beta$.

Note that SR is optimal when $\beta \to +\infty$, i.e. $s \to +\infty$. In this case, $s(SR)$ can be made arbitrarily close to -1. The asymptotic rate of convergence is then the same as for Fabian's algorithm, but [Polyak and Tsybakov, 1990] results comprise a larger class of functions: the class s of the function can be odd (whereas s has to be even in Theorem 2.3.2), and the sth derivative has to satisfy Hölder's condition, but it is not required for it to be bounded.

On the practical side, [Granichin, 2003] investigates optimal values of a and c (constants in the sequences $(a_n)_n$ and $(c_n)_n$).

In a setting related to Polyak-Tsybakov's algorithm and assumptions (see Theorem 2.3.4), [Bach and Perchet, 2016] provides additional results, generalizing Theorem 2.3.4. The optimization algorithm in [Bach and Perchet, 2016] follows the general framework of Polyak-Tsybakov's algorithm (Eqs. 2.14 and 2.22) but they average the recommendation, which is:

$$\tilde{x}'_n := \frac{1}{(n+1)} \sum_{k=0}^{n} \tilde{x}_k \quad \text{in the } \textit{unconstrained case,} \tag{2.24}$$

$$\tilde{x}'_n := \frac{2}{(n+1)(n+2)} \sum_{k=0}^{n} (k+1)\tilde{x}_k \quad \text{in the } \textit{constrained case.} \tag{2.25}$$

Especially, they provide results directly in terms of Simple Regret and specify the dependence in the dimension d. Their method relaxes the conditions on the kernel K (see Eq. 2.22). They provide explicit optimal values of a, α, c and γ (see Eqs. 2.14 and 2.22), depending on the order of differentiability s, the strong-convexity constant A_1, the 2nd-order smoothness A_2 and the dimension d (see Theorem 2.3.4). Notably, they distinguish:

- *convex* and *strongly-convex* functions, the latter corresponding to assumption 3 of Theorem 2.3.4. To the best of our knowledge, it is the first analysis of the convergence rate of an objective function only convex in the Kiefer-Wolfowitz optimization scheme;

- *constrained* and *unconstrained* optimization, the latter being as in the Polyak-Tsyakov setting with recommendation $\tilde{x}'_n$ (see Eq. 2.24) and the former being as Polyak-Tsybakov framework with an extra projection of $\tilde{x}_n$ on a compact convex set in Eq. 2.14 and recommendation $\tilde{x}'_n$ (see Eq. 2.25) instead of $\tilde{x}_n$;

- *asymptotic* and *non-asymptotic* regime, see Section 2.2.3.

Table 2.1: Upper bound on the Simple Regret using Polyak-Tsybakov's algorithm, analysed by [Bach and Perchet, 2016] in the non-asymptotic regime. Strongly convex function satisfies conditions 1-3 of Theorem 2.3.4. [Bach and Perchet, 2016] also provide results when the objective function is only convex.

Degree of smoothness s	Convex function	Strongly convex function
$s = 2$	$\left(\frac{d^2}{n}\right)^{1/3}$	$\sqrt{\frac{d^2}{A_1 n}}$
$s > 2$	$\left(\frac{d^2}{n}\right)^{\frac{s-1}{2s}}$	$\frac{1}{A_1^2}\left(\frac{d^2}{n}\right)^{\frac{s}{s+1}}$

The noise is such that $\mathbb{E}_\omega(f(x,\omega)) = f(x)$ and has a bounded variance. They found out that constrained or unconstrained optimization has no impact on the upper bound of the Simple Regret.

Under the conditions of Theorem. 2.3.4, they prove that, *asymptotically*, i.e. for n big enough, $\mathbb{E}(SR_n) = O\left(\frac{1}{A_1^2}(\frac{d^2}{n})^{\frac{s}{s+1}}\right)$. We recover $s(SR) = \frac{s}{s+1} \to 1$ when $s \to +\infty$ as in [Fabian, 1967] and [Polyak and Tsybakov, 1990]. Table 2.1 provides [Bach and Perchet, 2016]'s results in the non-asymptotic case, i.e. for a fixed n (possibly small).

Note that it is possible to approximate the gradient using only one call to the black-box per iteration, see e.g. [Granichin, 2003, Shamir, 2013, Polyak and Tsybakov, 1990], leading to similar convergence rates. Nonetheless, this method requires to project the recommendation $\tilde{x}_n$ obtained by Eq. 2.14 on a convex compact set containing x^* and an additional assumption on the objective function. We report also that most of the literature cited previously contains some results in term of limit distribution of the estimate.

Local regression algorithms

To the best of our knowledge, well tuned regression algorithms such as CLOP or QRL (see Section 2.1.3) can reach experimentally a rate $O(1/n)$ on smooth functions [Coulom, 2011, Coulom et al., 2011]. However, we are not aware of rigorous theoretical results.

2.3.2 Comparison-based algorithms

Among the algorithms designed to tackle the black-box noisy optimization problem, we find also the family of *population-based algorithms*. A population-based algorithm generates, at each iteration, a population of search points - or individuals - and then selects an approximation for the optimization problem. The selection is in general performed thanks to the ranking or comparisons between the search points, but not the direct use of the fitness values. In this last case, the algorithm is termed *comparison-based algorithm* or *Fitness Value Free*. Especially, the comparison-based feature confers some robustness. For example, some very small or high values will not affect the optimization process. A key point is that a comparison-based algorithm is invariant to translation, rotations and strictly increasing transformations[6] [Auger, 2016, Chotard, 2015]. Incidentally, some non-convex or some non-smooth functions can be optimized as easily as convex ones. Typically optimizing the composition $x \in \mathbb{R}^d \mapsto g(\|x - x^*\|^2)$ is as easy as optimizing the sphere for comparison-based algorithms as soon as g is increasing, even if g is non-differentiable or if the composed function is not convex. Evolutionary Algorithms are a subset of comparison-based algorithms where the population is generated randomly, inspired by biological processes.

One can also choose this family of algorithms by necessity. As a refinement of the black-box problem, we might encounter some optimization problems where the fitness value itself is unknown. In this case, an oracle only provides a ranking of a given set of points, but not the fitness values of these points. For example in games, an operator can compare two agents, but not directly provide a level evaluation. In design, with the human in the loop, a user preference is a comparison between two search points. Searching a Pareto front might also involve a user providing her preferences. In the noisy optimization setting, some *misranking* of individuals might result in poor recommendations. If we consider 2 individuals x_1 and x_2, due to the noise perturbation, we might obtain $f(x_1, \omega_1) > f(x_2, \omega_2)$ whereas actually the real ordering between the individuals is the opposite, i.e. $f(x_1) < f(x_2)$.

We review some families of comparison-based algorithms. We are particularly interested in Evolution Strategies, their convergence rates being already well studied in the noise-free case. Among the other comparison-based algorithms for continuous noisy optimization, we find e.g. Differential Evolution or Particle Swarm Optimization for which very few is know theoretically in terms of UR

[6] i.e. the algorithm behave the same on the original objective function and on the composition of this objective function with a translation, a rotation or a strictly increasing function.

or regrets. On the contrary, the behaviour of ES in the noise-free case has been investigated through the use of Markov Chain Theory (see Chap. 3).

Evolution Strategy (ES)

Vanilla Algorithm. Evolution Strategies are one of the very first Evolutionary Algorithms, introduced by [Rechenberg, 1965]. A specific $(1,\lambda)$-Self Adaptive Evolution Strategy is presented in Algorithm 2.1 and variants of ES are discussed below.

Algorithm 2.1 $(1,\lambda)$-Self Adaptive Evolution Strategy. The point(s) used to generate the offspring, here only x_{n-1}, is/are called *parent(s)*.

Input:
population size $\lambda \in \mathbb{N}$; real $\tau > 0$; initial recommendation x_0; initial step-size σ_0; Gaussian random vector $\mathscr{G}$; objective function f.

Output:
an approximation of the optimum x^* of the objective function f

$n \leftarrow 1$
while not finished **do**
  $\forall i \in \{1,\dots,\lambda\},\ \sigma_{n-1}^i \leftarrow \sigma_{n-1}\exp(\tau\mathscr{G})$ ▷ Step-size
  $\forall i \in \{1,\dots,\lambda\},\ x_n^i \leftarrow x_{n-1} + \sigma_{n-1}^i\mathscr{G}$ ▷ Offspring
  Rank $x_n^{i_1},\dots,x_n^{i_\lambda}$ such that $f(x_n^{i_1}) \leq f(x_n^{i_2}) \leq \cdots \leq f(x_n^{i_\lambda})$ ▷ Evaluation step
  $x_n \leftarrow x_n^{i_1}$ ▷ Selection step
  $\sigma_n \leftarrow \sigma_n^{i_1}$ ▷ Update Step-size
  $n \leftarrow n+1$
end while
  return x_n

In the black-box setting, the offspring are generally generated through the use of a normal distribution as in Alg. 2.1. However in the gray-box setting (e.g. separability of the objective function) other distributions (e.g. Cauchy distribution) are relevant, see [Hansen et al., 2006]. Instead of one parent as in Alg. 2.1, another classical variant - denoted (μ,λ)-ES - is to keep a population of μ parents for each generation. The recommendation is then the best offspring or a weighted recombination of these points. (μ,λ)-ES is called non-elitist. Elitist Evolution Strategy, denoted $(\mu+\lambda)$-ES, is an ES version in which, during the evaluation step, the λ offspring are ranked together with the μ parents, and the selection is

performed on these $\mu+\lambda$ points. Many works are devoted to the search of optimal population sizes λ and μ [Beyer, 2001]. A key point to obtain an efficient ES is to adapt the step size σ (said self-adaptive in Alg. 2.1) and the covariance matrix of the Gaussian vector $\mathscr{G}$. We find a rich literature on this topic [Hansen et al., 2015] - see notably the Cumulative Step-size Adaptation - Evolution Strategy (CSA-ES) and the Covariance Matrix Adaptation - Evolution Strategy (CMA-ES) methods. [Arnold and Beyer, 2006] experimentally shows that an ES without any adaptation to the additive noisy setting stagnates at some distance of the optimum, it does not converge.

"Mutate Large, But Inherit Small". [Rechenberg, 1994, Beyer, 1998] have proposed variants of Evolution Strategy using large mutations and small inheritance. Under specific conditions, when the variance of the noise decreases to zero in the vicinity of the optimum, the dynamics[7] of such a modified algorithm *asymptotically* approaches the performance of a classical ES without noise. It is asymptotic since it is only when the dimension goes toward infinity.

Resampling for ES. We already discussed the different methods to deal with noisy objective functions in Section 2.1.3. We develop here the resampling technique. When using an Evolution Strategy in a context with additive Gaussian noise, [Astete-Morales et al., 2014] has shown mathematically that an exponential number of resamplings (number of resamplings scaling exponentially with the index of iterations) or an adaptive number of resamplings (scaling as a polynomial of the inverse step-size) can both lead to a log-log convergence rate with high probability for the Simple Regret over the objective functions of the form $x \mapsto \|x-x^*\|^p$, with p a positive integer. [Astete-Morales et al., 2014] experimentally showed that a polynomial number of resamplings (number of resamplings scaling polynomially with the index of iterations) also leads to a log-logarithmic rate of convergence with high probability for *SR* over the same family of objective functions.

Uncertainty Handling - Covariance Matrix Adaptation - Evolution Strategy (UH-CMA-ES). [Hansen et al., 2009] developed a specific variant of the CMA-ES [Hansen and Ostermeier, 2001] to handle uncertainty. More specifically, it uses an adaptive number of resamplings in order to reduce the noise. It combines the traditional CMA-ES algorithm with an Uncertainty-Handling tool. The

[7] dynamics in term of progress rate, which is not detailed here.

Uncertainty-Handling tool is made of two parts. The first part *measures* the uncertainty due to the noise and the second part *handles* the uncertainty. The treatment of the uncertainty is twofold. If the measurement of the uncertainty exceeds a given threshold, then *the computation time* (typically the number of resamplings) increases and/or the variance (the step-size) of the population increases. Whereas if the uncertainty is below the threshold, the *computation time* decreases. UH-CMA-ES was successfully applied to the online optimization of feedback controllers of thermoacoustic instabilities of gas turbine combustors. To the best of our knowledge, there are no theoretical results regarding the convergence of UH-CMA-ES. However, it seems to perform poorly when confronted with high levels of noise [Coulom et al., 2011].

It results from above that the best ES adaptations to additive noise are able to reach a log-log converge for the *SR*. However, we are not aware of any known bounds (upper or lower) on this rate of convergence.

Bernstein races.

Some methods are based on Hoeffding or Bernstein inequalities in order to find adaptively the number of resampling necessary to distinguish between several points at each iteration [Mnih et al., 2008].

Using this method, [Coulom et al., 2011] handled the specific case of Bernoulli random variables. Their algorithm, termed Racing-based Estimation of Distribution Algorithms (R-EDA), reaches a *UR* in $O(1/n^{1/p})$ with high probability up to logarithmic factor when the objective function is $x \mapsto \lambda \|x - x^*\|^p$. The rate is $O(1/n^{1/2p})$ when the sphere is translated: $x \mapsto \lambda \|x - x^*\|^p + c$. Even if R-EDA is suboptimal from a theoretical point of view (see discussion about lower bound below), it is in practice quite efficient.

2.3.3 General lower bound

The fastest algorithms described previously have a *SR* in $O(n^{-\frac{s}{s+1}})$, where s is the order of differentiability of the objective function. [Chen, 1988] has proved that it is actually the optimal rate of convergence, see Theorem 2.3.5 below.

Theorem 2.3.5 (Chen's lower bound)**.** *Let I be a bounded interval of $\mathbb{R}$. Assume that the querying algorithm satisfies:*

- *the nth search point $x_n \in I$ is a computable function of the previous search points $x_1, \ldots, x_{n-1}$ and their corresponding evaluations $y_1, \ldots, y_{n-1}$, and*

only of these data;

- *each estimate $\tilde{x}_n$ is a function (possibly randomized) of $x_1,\dots,x_n$ and $y_1,\dots,y_n$.*

Assume that the noisy estimation y_n has a conditional density equal to $g(\cdot,x_n,f(x_n))$ with respect to the Lebesgue measure μ, where $g(u,v,w)$ is such that:

- *g is positive;*
- *g' and g'', derivatives with respect to w exist $\forall\ w \in \mathbb{R}$;*
- *g'' and $(\log g)''$ (derivatives with respect to w) are integrable $\forall\ v,w$.*

Let $s > 1$ be odd and $\delta_0 > 0$. Assume that $\mathscr{F}$ is a family of objective functions on I satisfying:

- *$\forall f \in \mathscr{F}$, $x^*(f)$ is a point of global maximum of f;*
- *$\forall \delta \in [0,\delta_0]$, $\forall\ n \in \mathbb{N}^*$, the function*

$$f_{n\delta}(x) = -x^* + 2\delta n^{-1/2} arctan(n^{1/2s}(x - \delta n^{-(s-1)/(2s)})) \tag{2.26}$$

is in $\mathscr{F}$.

Then $\forall \eta \in (0,1)$, $\exists c > 0$ such that $\forall n$,

$$\inf_{f\in\mathscr{F}} \mathbb{P}(|\tilde{x}_n - x^*(f)|^2 \geq cn^{-(s-1)/s}) \geq \eta. \tag{2.27}$$

In particular, for δ sufficiently small, $f_{n\delta}$ is concave. [Chen, 1988] also states that this result can be extended to functions in I^d and to infinite intervals I.

[Polyak and Tsybakov, 1990] widened this results to a larger class of functions.

Theorem 2.3.6 (Polyak-Tsybakov's lower bound)**.** *Let $\tilde{x}_n$ be any Borel function of $x_1,\dots,x_n,y_1,\dots,y_n$ where:*

- *$x_1 = h_1(\zeta)$, with ζ a random variable of arbitrary probability $\mathbb{P}_\zeta$ and h_1 a measurable function;*
- *$\forall\ i \in \{2,\dots,n\}$, $x_{i+1} = h_i(x_1,\dots,x_i,y_1,\dots,y_i,\zeta)$, with ζ a random variable of arbitrary probability $\mathbb{P}_\zeta$ and h_i a measurable function;*

- $\forall\, i \in \{2,\dots,n\}$, $y_i = f(x_i) + \omega_i$, *with* $\omega_1,\dots,\omega_n$ *identically distributed with the distribution function* g *such that*

$$\int \ln(dg(u)/dg(u+t))dg(u) \leq I_0 t^2 \quad |t| \leq t_0, \tag{2.28}$$

  for some $0 < t_0 \leq \infty$ *and* $0 < I_0 < \infty$.

Let $\mathscr{F}$ *be the family of functions as in Theorem 2.3.4, i.e.,* $\forall\, f \in \mathscr{F}$, f *satisfies:*

- f *has a unique optimum at* $x^*(f)$*;*
- f *has continuous partial derivatives up to order* s *inclusive, which satisfy the Hölder condition of order* $\alpha \in (0,1]$*;*
- $\forall\, x \in \mathbb{R}^d$, $(Df(x), x - x^*) \geq A_1 \|x - x^*\|^2$*;*
- $\forall\, x, x' \in \mathbb{R}^d$, $\|Df(x) - Df(x')\| \leq A_2 \|x - x'\|$,

where A_1 *and* A_2 *are finite positive constants,* $A_2 > A_1$ *and* $\beta = s + \alpha \geq 2$. *Then*

$$\inf_n \inf_{\tilde{x}_n} \sup_{f \in \mathscr{F}} n^{(\beta-1)/\beta} \mathbb{E}(\|\tilde{x}_n - x^*(f)\|^2) > 0. \tag{2.29}$$

In particular, this lower bound holds for the family of strongly convex objective functions with partial derivatives smooth enough, and contrarily to Chen, β can be odd or even.

When the noise is additive with bounded variance, [Jamieson et al., 2012] exhibited a lower bound in $\Omega(n^{-1/2})$ on the family of strongly convex functions with Lipschitz gradient on a convex subset of $\mathbb{R}^d$. However this is not in contradiction with Chen and Polyak-Tsybakov's results, as the setting in [Jamieson et al., 2012] is slightly different: it allows queries for search points only at distance $O(n^{-1/4})$ of the optimum.

[Shamir, 2013] recovers this lower bound $\Omega(n^{-1})$ in the case of the quadratic functions, and a bound $\Omega(n^{-1/2})$ for the family of strongly convex and smooth functions. Here again, this might appear as a contradiction with Chen and Polyak-Tsybakov's bounds, however, the setting in [Shamir, 2013] is non-asymptotic. We discuss this difference in Appendix B.

On the other hand, [Shamir, 2013] shows that the optimal expected CR for quadratic, strongly convex and smooth objective functions is in $\Omega(n^{1/2})$, a bound which is reached by various algorithms [Fabian, 1967, Shamir, 2013].

In the specific case of the Bernoulli noise, [Rolet and Teytaud, 2010b] showed that for every monotonic transformation of the sphere function, $UR_n = \Omega(1/n)$ with high probability.

2.4 Contributions in noisy black-box continuous optimization

2.4.1 Comparison-based algorithms

Evolution Strategies & small noise. [Jebalia et al., 2011] studied the case of multiplicative noise for ES. Chapter 3 investigates the case of a noise decreasing faster than the multiplicative setting (i.e. $z > 1$ in Eq. 2.7). We show that by slightly modifying a classical ES - adding some resamplings - we can recover the same rate of convergence of a classical ES in the noise-free case.

Evolution Strategies & additive noise: a lower bound. Several adaptations of Evolution Strategies have been proposed to tackle the additive noise problem. Theoretical and experimental results show a log-log convergence of the Simple Regret. However, we are not aware of some bounds on this convergence rate $s(SR)$. This question will be addressed in Chapter 4, where we prove that $s(SR) \geq -1/2$ in expectation for some algorithms fitting a specific framework. In particular, the theoretical study covers a wide family of ESs - but not all - and possibly other optimization algorithms.

Comparison-based algorithms & additive noise: an upper-bound. A natural question that arises is then to know if comparison-based algorithms can be as fast as value-based algorithms (which can get $s(SR) \to -1$) in the additive noise setting. We provide in Chapter 5 a comparison-based algorithm performing as fast as a value-based algorithm. That is, it might be possible for an ES to be fast when confronted with big noise, however it must satisfy some particular design in the mutation step.

2.4.2 Value-based algorithms and small noise

Performances of value-based algorithms, such as Kiefer-Wolfowitz algorithms, are well known in the context of additive noise. Chap. 6 describes a generic framework which covers different optimization algorithms - among them Kiefer-Wolfowitz algorithms but also Newton's algorithms (using an approximate Hessian) - as well as various strength of noise, from additive to multiplicative case. Concerning additive noise, we recover the results previously proven. We extend these results to smaller strengths of noise.

2.4.3 Right algorithm? Right parameters?

It already appeared in the literature review that the performances of an algorithm depend on:

- the noise: additive, multiplicative, ...;
- the objective function: differentiable, smooth, invariant to translations, ...;
- the starting point: first search point(s) near to the global optimum or near to a local optimum;
- the parameters of the algorithm: budget, step-size, ...;
- ...

When facing a noisy black-box optimization problem, without any idea or intuition of the objective function properties and/or of the model of noise, which algorithm should one use? and how should the parameters be set? Chap. 7 presents a solution under the form of a *portfolio*: we can set different algorithms in the portfolio: an ES, a Fabian's algorithm, ... This method guarantees to provide an approximation of the optimum following the convergence rate of the best of the algorithms contained in the portfolio. In particular, we can set several times the same algorithm with different parametrizations or even several times the same algorithm with the same parametrization to avoid a 'bad luck' starting point. Portfolios in the noise-free case are well studied. However, we will see that in the noisy setting, the selection among the different optimization algorithms is far from being straightforward.

Chapter 3

Evolution Strategies confronted with small noise

This chapter is based on:

Cauwet, M. (2014). Noisy optimization: Convergence with a fixed number of resamplings. In *Applications of Evolutionary Computation - 17th European Conference, EvoApplications 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers*, pages 603–614

As discussed in Chapter 2, Evolution Strategies in continuous domains might not converge in the presence of noise. Under mild assumptions, and using an increasing number of resamplings, one can mitigate the effect of additive noise and recover a logarithmic convergence:

$$\limsup_n \frac{\log \|\tilde{x}_n - x^*\|}{\log n} = -A \ < \ 0 \qquad \text{a.s.} \tag{3.1}$$

Robustness of ESs confronted with multiplicative noise - without any need of an adaptation scheme - has also been investigated. We study in this chapter the model of noise for which variance decreases around the optimum slightly faster than in the multiplicative noise model. This noise model is described in Eq. 2.7 with $z > 1$. We recall it:

$$f(x,\omega) = f(x) + (f(x) - f(x^*))^z \omega. \tag{2.7}$$

We assume that ω has a zero-mean and a bounded variance, that $f(x^*) = \inf_{x\in\mathscr{D}} f(x) = 0$, so that[1] $Var(f(x,\omega)) = O(\mathbb{E}f(x,\omega)^{2z})$; w.l.o.g, we assume that $Var(\omega) \leq 1$ and $Var(f(x,\omega)) \leq (\mathbb{E}f(x,\omega))^{2z}$. We show new sufficient conditions for the convergence of an Evolution Strategy with a constant number of resamplings in this noise setting. In particular, we get faster rates - log-linear convergence - than in the case of additive noise:

$$\limsup_n \frac{\log \|\tilde{x}_n - x^*\|}{n} = -A \ < \ 0 \qquad \text{a.s.} \tag{3.2}$$

3.1 Log-linearity

3.1.1 Preliminary: noise-free case

We recall that a $(1,\lambda)$-Evolution Strategy at iteration n:

- generates λ individuals using the current estimate $\tilde{x}_{n-1}$ of the optimum x^* and the so-called mutation strength (or step-size) σ_{n-1};
- provides a pair $(\tilde{x}_n, \sigma_n)$ where $\tilde{x}_n$ is a new estimate of x^* and σ_n is a new mutation strength.

From now on, for the sake of notation simplicity, we assume that $x^* = 0$.

For some ESs and in the noise-free case, we know (see e.g. Theorem 4 in [Auger, 2005]) that there exists a constant A such that :

$$\frac{\log(\|x_n\|)}{n} \xrightarrow[n\to\infty]{a.s} -A \tag{3.3}$$

$$\frac{\log(\sigma_n)}{n} \xrightarrow[n\to\infty]{a.s} -A \tag{3.4}$$

This chapter discusses cases in which an ES with a convergence rate as in Eqs. 3.3 and 3.4 in the noise-free case also has this convergence rate in a noisy setting.

In the general case of arbitrary ESs, we don't know if A is positive. However, through the study of an underlying Markov chain, it has been possible to show that:

[1] In the present document, we slightly modified the modelization of [Cauwet, 2014] in order to fit the general model of Eq. 2.7. Hence the reader should not be confused by the shift of a factor 2 in all the chapter compared to the original paper.

- for a $(1+1)$-ES with generalized one-fifth success rule applied to scale invariant positively homogeneous functions, $A > 0$, [Auger and Hansen, 2013];
- for a $(1,\lambda)$-Self Adaptive Evolution Strategy (SA-ES) with Gaussian mutations to optimize the sphere function, the estimate of A by Monte-Carlo simulations is positive [Auger, 2005].
- in the case of a $(1,\lambda)$-CSA-ES on linear functions, it was shown in [Chotard et al., 2012] that if $\lambda \geq 3$ and $c = 1$[2], then $A < 0$ and the algorithm diverges log-linearly. Furthermore, the rate A is provided explicitly.

Property 3.1.1. *For some $\delta > 0$, for any α, α' such that $\alpha < A$ and $\alpha' > A$, there exist $C > 0$, $C' > 0$, $V > 0$, $V' > 0$, such that with probability at least $1-\delta$*

$$\forall n \geq 1, C' \exp(-\alpha' n) \leq \|x_n\| \leq C \exp(-\alpha n); \tag{3.5}$$

$$\forall n \geq 1, V' \exp(-\alpha' n) \leq \sigma_n \leq V \exp(-\alpha n). \tag{3.6}$$

Proof. For any $\alpha < A$, almost surely, $\log(\|x_n\|) \leq -\alpha n$ for n sufficiently large. So, almost surely, $\sup_{n\geq 1} \log(\|x_n\|) + \alpha n$ is finite. Consider V the quantile $1-\frac{\delta}{4}$ of $\exp\left(\sup_{n\geq 1} \log(\|x_n\|) + \alpha n\right)$. Then, with probability at least $1-\frac{\delta}{4}$, $\forall n \geq 1$, $\|x_n\| \leq V \exp(-\alpha n)$. We can apply the same trick for lower bounding $\|x_n\|$, and upper and lower bounding σ_n, all of them with probability $1-\frac{\delta}{4}$, so that all bounds hold true simultaneously with probability at least $1-\delta$. □

3.1.2 Noisy case

If some Evolution Strategies converge log-linearly as in Eqs. 3.3 and 3.4 in the noise free-case, then just by considering Y resamplings for each search point as explained in Alg. 3.1, they will also be fast in the noisy case. This is formalized in Theorem 3.1.1.

Our theorem holds for any Evolution Strategy satisfying the following constraints:

- At each iteration n, a search point x_n is defined and λ search points are generated and have their fitness values evaluated.
- The noisy fitness values are averaged over Y (a constant) resamplings.

[2] specific parameter of CSA-ES

- The jth individual evaluated at iteration n is randomly drawn by $x_n + \sigma_n \mathcal{G}_d$ with $\mathcal{G}_d$ a d-dimensional standard Gaussian variable.

This framework is presented in Alg. 3.1.

Theorem 3.1.1. *Consider the following assumptions:*

(i) the fitness function f satisfies $\mathbb{E}f(x,\omega) = \|x\|^p$ and has a limited variance:

$$Var(f(x,\omega)) \leq (\mathbb{E}f(x,\omega))^{2z} \text{ for some } z > 1; \tag{3.7}$$

(ii) in the noise-free case, the ES with population size λ under consideration is log-linearly converging, i.e. for any $\delta > 0$, for some $\alpha > 0$, $\alpha' > 0$, there exist $C > 0$, $C' > 0$, $V > 0$, $V' > 0$, such that with probability 1-δ, Eqs. 3.5 and 3.6 hold;

(iii) the number Y of resamplings per individual is constant.

Then, if $z > \max\left(\frac{p\alpha' - (\alpha-\alpha')d}{p\alpha}, \frac{2\alpha'-\alpha}{\alpha}\right)$, for any $\delta > 0$, there is $Y_0 > 0$ such that for any $Y \geq Y_0$, Eqs. 3.5 and 3.6 also hold with probability at least $(1-\delta)^2$ in the noisy case.

Corollary 3.1.1. *Under the same assumptions, with probability at least $(1-\delta)^2$,*

$$\limsup_n \frac{log(\|\tilde{x}_n\|)}{n} \leq -\frac{\alpha}{\lambda Y}$$

Proof of Corollary 3.1.1. Immediate consequence of Theorem 3.1.1, by applying Eq. 3.5 and using $\limsup_n \frac{log(\|\tilde{x}_n\|)}{n} = \limsup_n \frac{log(\|x_n\|)}{\lambda Y n}$. □

Informally speaking, our theorem shows that if an algorithm converges in the noise-free case, then it also converges in the noisy case with the resampling rule, at least if z and Y are large enough. We show a log-linear convergence rate as in the noise-free case. This means that we get $\log\|\tilde{x}_n\|$ linear in the number of function evaluations. This is as Eq. 3.2, and faster than Eq. 3.1 which is typical for noisy optimization with constant variance.

Notice that we can choose constants α and α' very close to each other. Then the assumption $z > \max\left(\frac{p\alpha' - (\alpha-\alpha')d}{p\alpha}, \frac{2\alpha'-\alpha}{\alpha}\right)$ is equivalent to $z > 1$.

Algorithm 3.1 A general framework for Evolution Strategies. For simplicity, it does not cover all Evolution Strategies, e.g. mutations of step-sizes as in self-adaptive algorithms are not covered; yet, our proof can be extended to a more general case ($x_{n,i}$ distributed as $x_n + \sigma\mathscr{G}$ for some noise $\mathscr{G}$ with exponentially decreasing tail). The case $Y = 1$ is the case without resampling.

Input:

Population size λ; initial recommendation $\tilde{x}_0$; initial step-size σ_0; Gaussian random vector $\mathscr{G}$, objective function f.

Output:

an approximation of the optimum x^* of the objective function f

```
n ← 1
while not finished do
    for i ∈ {1,...,λ} do
        x_{n,i} ← x̃_n + σ_n G_d.
        y_{n,i} ← (1/Y) Σ_{k=1}^{Y} f(x_{n,i}, ω_k).
    end for
    (x̃_{n+1}, σ_{n+1}) ← update(x_{n,1},...,x_{n,λ}, y_{n,1},...,y_{n,λ}, σ_n).
    n ← n + 1
end while
    return x̃_n
```

In the previous hypothesis, the new individuals are drawn following $\tilde{x}_n + \sigma_n\mathscr{G}_d$ with $\mathscr{G}_d$ a d-dimensional standard Gaussian variable, but we could substitute $\mathscr{G}_d$ for any random variable with an exponentially decreasing tail.

Sketch of proof: Consider an arbitrary $\delta > 0$ and $\delta_n = \exp(-\gamma n)$ for some $n \geq 1$ and $\gamma > 0$. Lemma 3.1.2 gives the probability that at least two generated points x_{n,i_1} and x_{n,i_2} at iteration n are "close", i.e. are such that $\mid \|x_{n,i_1}\|^p - \|x_{n,i_2}\|^p \mid \leq \delta_n$; then the probability that the noise of at least one of the λ evaluated individuals of iteration n is bigger than $\frac{\delta_n}{2}$ is provided in Lemma 3.1.3. Thus, we can conclude in Lemma 3.1.4 by estimating the probability that at least two individuals are misranked due to noise.
We first begin by showing a technical lemma.

Lemma 3.1.1. *Let $u \in \mathbb{R}^d$ be a unit vector and $\mathscr{G}_d$ a d-dimensional standard normal random variable. Then for $S > 0$ and $\ell > 0$, there exists a constant $M > 0$*

such that :

$$\max_{v\geq 0}\mathbb{P}(|\ \|u+S\mathscr{G}_d\|^p-v|\leq \ell)\leq MS^{-d}\max\left(\ell,\ell^{d/p}\right).$$

Proof of Lemma 3.1.1. For any $v\geq \ell$, we denote $E_{v\geq\ell}$ the set :

$$E_{v\geq\ell}=\{x\,;|\ \|x\|^p-v\ |\leq \ell\}=\left\{x\,;(v-\ell)^{\frac{1}{p}}\leq \|x\|\leq (v+\ell)^{\frac{1}{p}}\right\}.$$

We first compute $\mu(E_{v\geq\ell})$, the Lebesgue measure of $E_{v\geq\ell}$:

$$\mu(E_{v\geq\ell})=K_d\left\{(v+\ell)^{\frac{d}{p}}-(v-\ell)^{\frac{d}{p}}\right\},$$

with $K_d=\frac{(2\pi)^{d/2}}{2\times 4\times\cdots\times d}$ if d is even, and $K_d=\frac{2(2\pi)^{(d-1)/2}}{1\times 3\times\cdots\times d}$ otherwise. Hence, by Taylor expansion, $\mu(E_{v\geq\ell})\leq Kv^{\frac{d}{p}-1}\ell$, where $K=K_d\left(2\frac{d}{p}+\sup_{v\geq\ell}\sup_{0<\zeta<\frac{\ell}{v}}\frac{q''(\zeta)}{2}\frac{\ell}{v}\right)$, with $q(x)=(1+x)^{\frac{d}{p}}$.

• If $v\geq\ell$:

$$\begin{aligned}\mathbb{P}(|\ \|u+S\mathscr{G}_d\|^p-v|\leq\ell) &= \mathbb{P}(u+S\mathscr{G}_d\in E_{v\geq\ell}),\\ &\leq S^{-d}\sup_{x\in E_{v\geq\ell}}\left(\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{\|S^{-1}(x-u)\|^2}{2}\right)\right)\mu(E_{v\geq\ell}),\\ &\leq M_1S^{-d}\ell,\\ &\leq M_1S^{-d}\max\left(\ell,\ell^{d/p}\right),\end{aligned}$$

where $M_1=\frac{K}{\sqrt{2\pi}}\sup_{v\geq\ell}\sup_{x:\|x\|\leq(v+\ell)^{\frac{1}{p}}}\left[v^{\frac{d}{p}-1}\exp\left(-\frac{\|S^{-1}(x-u)\|^2}{2}\right)\right]$.

• If $v<\ell$, $\quad\mathbb{P}(|\ \|u+S\mathscr{G}_d\|^p-v|\leq\ell)\leq M_2S^{-d}\ell^{d/p}\leq M_2S^{-d}\max\left(\ell,\ell^{d/p}\right)$,

where $M_2=2^{\frac{d}{p}}\frac{K_d}{\sqrt{2\pi}}$. Hence the result follows by taking $M=\max(M_1,M_2)$. □

Lemma 3.1.2. *Let us denote by $P_n^{(1)}$ the probability that, at iteration n, there exist at least two points x_{n,i_1} and x_{n,i_2} such that $|\ \|x_{n,i_1}\|^p-\|x_{n,i_2}\|^p\ |\leq\delta_n$. Then*

$$P_n^{(1)}\leq B\lambda^2\exp(-\gamma' n),$$

for some $B>0$ and $\gamma'>0$ depending on γ, d, p, C, C', V, α, α'.

Proof of Lemma 3.1.2. Let us first compute the probability $P_n^{(0)}$ that, at iteration n, two given generated points x_{n,i_1} and x_{n,i_2} are such that $|\ \|x_{n,i_1}\|^p - \|x_{n,i_2}\|^p\ | \leq \delta_n$. Let us denote by $\mathscr{G}_d^1$ and $\mathscr{G}_d^2$ two d-dimensional standard independent random variables, $u \in \mathbb{R}^d$ a unit vector and $S_n = \frac{\sigma_n}{\|x_n\|}$.

$$\begin{aligned} P_n^{(0)} &= \mathbb{P}\left(|\ \|x_n + \sigma_n \mathscr{G}_d^1\|^p - \|x_n + \sigma_n \mathscr{G}_d^2\|^p\ | \leq \delta_n\right), \\ &= \mathbb{P}\left(|\ \|u + S_n \mathscr{G}_d^1\|^p - \|u + S_n \mathscr{G}_d^2\|^p\ | \leq \frac{\delta_n}{\|x_n\|^p}\right), \\ &\leq \max_{v \geq 0} \mathbb{P}\left(|\ \|u + S_n \mathscr{G}_d^1\|^p - v| \leq \frac{\delta_n}{\|x_n\|^p}\right). \end{aligned}$$

Hence, by Lemma 3.1.1, there exists a $M > 0$ such that $P_n^{(0)} \leq M S_n^{-d} \left(\frac{\delta_n}{\|x_n\|^p}\right)^m$, where m is such that $\left(\frac{\delta_n}{\|x_n\|^p}\right)^m = \max\left(\frac{\delta_n}{\|x_n\|^p}, \left(\frac{\delta_n}{\|x_n\|^p}\right)^{d/p}\right)$. Moreover $S_n \geq V' C^{-1} \exp(-(\alpha' - \alpha)n)$ by Assumption (ii). Thus $P_n^{(0)} \leq B \exp(-\gamma' n)$, with $B = M V'^{-d} C^d C'^{-mp}$ and $\gamma' = d(\alpha - \alpha') + m\gamma - mp\alpha'$. In particular, γ' is positive, provided that γ is sufficiently large. By union bound, $P_n^{(1)} \leq \frac{(\lambda-1)\lambda}{2} P_n^{(0)} \leq B\lambda^2 \exp(-\gamma' n)$. □

We now provide a bound on the probability $P_n^{(3)}$ that the fitness value of at least one search point generated at iteration n has noise (i.e. deviation from expected value) bigger than $\frac{\delta_n}{2}$ in spite of the Y resamplings.

Lemma 3.1.3.

$$P_n^{(3)} := \mathbb{P}\left(\exists i \in \{1, \ldots, \lambda\}\ ;\ \left|\frac{1}{Y}\sum_{j=1}^{Y} f(x_{n,i}, \omega_j) - \mathbb{E} f(x_{n,i}, \omega_j)\right| \geq \frac{\delta_n}{2}\right)$$

$$\leq \lambda B' \exp(-\gamma'' n)$$

for some $B' > 0$ and $\gamma'' > 0$ depending on γ, d, p, z, C, Y, α, α'.

Proof of Lemma 3.1.3. First, for one point x_{n,i_0}, $i_0 \in \{1, \ldots, \lambda\}$ generated at iteration n, we write $P_n^{(2)}$ the probability that when evaluating the fitness function at this point, we make a mistake bigger than $\frac{\delta_n}{2}$.
$P_n^{(2)} = \mathbb{P}(|\frac{1}{Y}\sum_{j=1}^{Y} f(x_{n,i_0}, \omega_j) - \mathbb{E}\left[f(x_{n,i_0}, \omega_j)\right]| \geq \frac{\delta_n}{2}) \leq B' \exp(-\gamma'' n)$ by using

Chebyshev's inequality, where $B' = 4Y^{-1}C^{2pz}$ and $\gamma'' = \alpha 2zp - 2\gamma$. In particular, $\gamma'' > 0$ if $z > \frac{mp\alpha' - (\alpha-\alpha')d}{p\alpha m}$; hence, if $z \geq \max\left(\frac{p\alpha'-(\alpha-\alpha')d}{p\alpha}, \frac{2\alpha'-\alpha}{\alpha}\right)$, we get $\gamma'' > 0$. Then, $P_n^{(3)} \leq \lambda P_n^{(2)}$ by union bound. □

Lemma 3.1.4. *Let us denote by $P_{misranking}$ the probability that in at least one iteration, there is at least one misranking of two individuals. Then, if $z > \max\left(\frac{p\alpha'-(\alpha-\alpha')d}{p\alpha}, \frac{2\alpha'-\alpha}{\alpha}\right)$ and Y is large enough, $P_{misranking} \leq \delta$.*

Proof of Lemma 3.1.4. We consider the probability $P_n^{(4)}$ that two individuals x_{n,i_1} and x_{n,i_2} at iteration n are misranked due to noise, so

$$\|x_{n,i_1}\|^p \leq \|x_{n,i_2}\|^p \tag{3.8}$$

$$\text{and } \frac{1}{Y}\sum_{j=1}^{Y} f(x_{n,i_1},\omega_j) \geq \frac{1}{Y}\sum_{j=1}^{Y} f(x_{n,i_2},\omega_j) \tag{3.9}$$

Eqs. 3.8 and 3.9 occur simultaneously if either two points have very similar fitness (difference less than δ_n) or the noise is big (larger than $\frac{\delta_n}{2}$). Therefore, $P_n^{(4)} \leq P_n^{(1)} + P_n^{(3)} \leq \lambda^2 P_n^{(0)} + \lambda P_n^{(2)} \leq (B+B')\lambda^2 \exp(-\min(\gamma',\gamma'')n)$.
$P_{misranking}$ is upper bounded by $\sum_{n\geq 1} P_n^{(4)} < \delta$ if γ' and γ'' are positive and constants large enough. γ' and γ'' can be chosen positive simultaneously if $z > \max\left(\frac{p\alpha'-(\alpha-\alpha')d}{p\alpha}, \frac{2\alpha'-\alpha}{\alpha}\right)$. □

Proof of Theorem 3.1.1. This lemma implies that with probability at least $1-\delta$, provided that Y has been chosen large enough, we get the same rankings of points as in the noise free case. In the noise free case Eqs. 3.5 and 3.6 hold with probability at least $1-\delta$ - this proves the convergence with probability at least $(1-\delta)^2$, hence the expected result; the proof of the theorem is complete. □

3.2 Experiments: how to choose the right number of resampling?

We consider in our experiments a version of multi-membered Evolution Strategies, the (μ,λ)-ES, where μ denotes the number of parents and λ the number of offspring ($\mu \leq \lambda$; see Alg. 3.2). We denote $(x_n^1,\ldots,x_n^\mu)$ the μ parents at iteration n and $(\sigma_n^1,\ldots,\sigma_n^\mu)$ their corresponding step-size. At each iteration, a (μ,λ)-ES

noisy algorithm: (i) generates λ offspring by mutation on the μ parents, using the corresponding mutated step-size, (ii) selects the μ best offspring by ranking the noisy fitness values of the individuals. Thus, the current approximation of the optimum x^* at iteration n is x_n^1.

Experiments are performed on the fitness function $f(x,\omega) = \|x\|^p + \|x\|^{pz}\mathscr{G}$, with $x \in \mathbb{R}^{15}$, $p = 2$, $z = 1.05$, $\lambda = 4$, $\mu = 2$, and $\mathscr{G}$ a standard Gaussian random variable, using a budget of 500000 evaluations. The results presented here are the mean and the median over 50 runs. The positive results are proved above, in a slightly different setting, for a given quantile of the results. This explains the good performance in Fig. 3.1 (median result) as soon as the number of resamplings is enough. The median performance is optimal with just 12 resamplings. On the other hand, Fig. 3.2 shows the mean performance of Alg. 3.2 with various numbers of resamplings. We see that a limited number of runs diverge so that the mean results are bad even with 16 resamplings; results are optimal (on average) for 20 resamplings.

Results are safer with 20 resamplings (for the mean), but faster (for the median) with a smaller number of resamplings.

3.3 Conclusion

We have shown that applying Evolution Strategies with a finite number of resamplings when the noise in the function decreases quickly enough near the optimum provides a convergence rate as fast as in the noise-free case. More specifically, if the noise decreases slightly faster than in the multiplicative model of noise, using a constant number of revaluation leads to a log-linear convergence of the algorithm. The limit case of a multiplicative noise has been analyzed in [Jebalia et al., 2011] where they consider a scale-invariant $(1+1)$-ES; a fixed number of resamplings is not sufficient for convergence when the noise is unbounded.

Further work. We did not provide any hint for choosing the number of resamplings. Proofs based on Bernstein races [Mnih et al., 2008] might be used for adaptively choosing the number of resamplings.

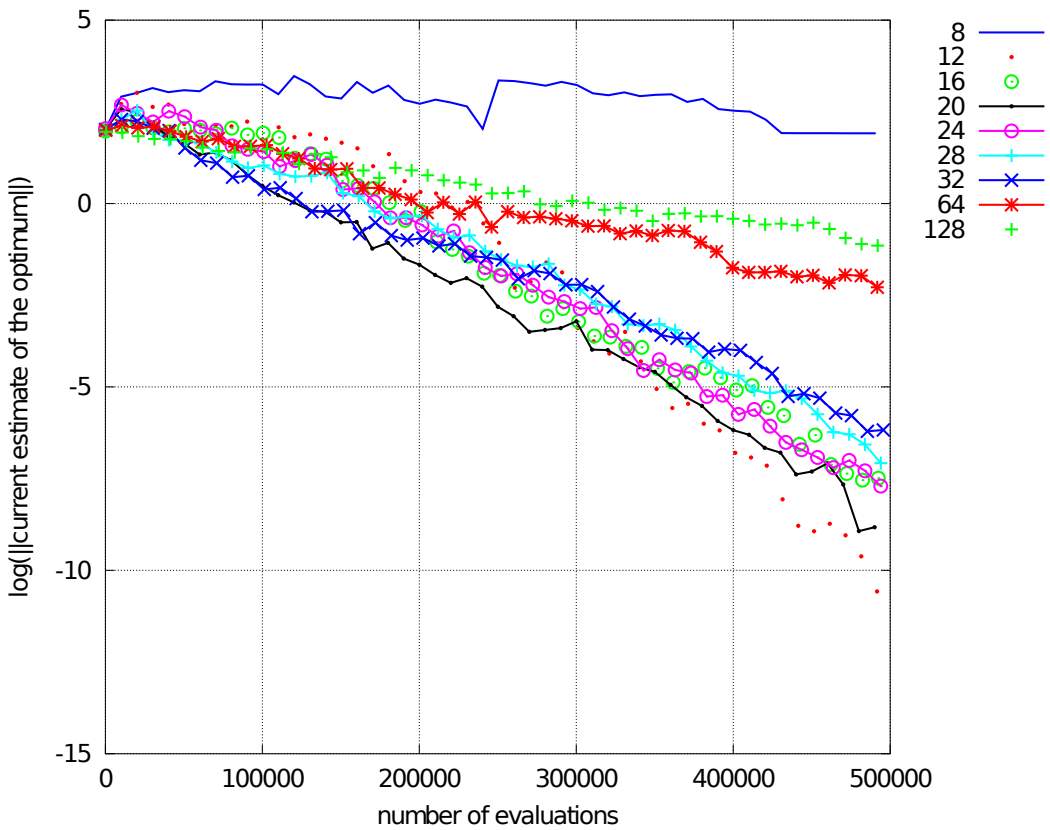


Figure 3.1: Convergence of Self Adaptive Evolution Strategy: Median results. Dimension $d = 15$, $p = 2$, $z = 1.05$, $\lambda = 4$, $\mu = 2$, over 50 runs.

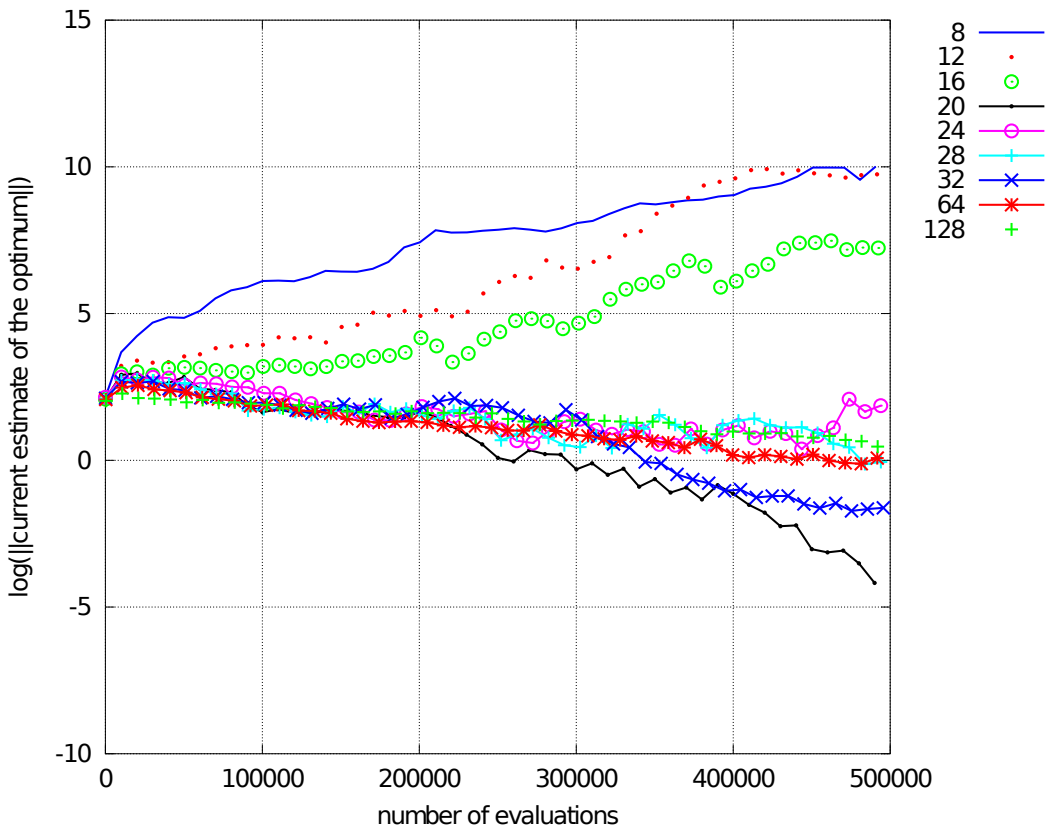


Figure 3.2: Convergence of Self Adaptive Evolution Strategy: Mean results. Dimension $d = 15$, $p = 2$, $z = 1.05$, $\lambda = 4$, $\mu = 2$, over 50 runs.

Algorithm 3.2 An Evolution Strategy, with constant number of resamplings. If we consider $Y = 1$, we obtain the case without resampling. $\mathscr{G}_k$ is a k-dimensional standard normal random variable.

Input:

number of resamplings $Y > 0$; $\lambda \geq \mu > 0$; a dimension $d > 0$; μ initial points $x_1^1, \ldots, x_1^\mu \in \mathbb{R}^d$ and initial step size $\sigma_1^1 > 0, \ldots, \sigma_1^\mu > 0$; Gaussian random vector $\mathscr{G}$; objective function f

Output:

an approximation of the optimum x^* of the objective function f

---

$n \leftarrow 1$
while not finished **do**
    Generate λ individuals independently using :

$$\begin{aligned} \sigma_j &\leftarrow \sigma_n^{((j-1) \ mod \ \mu)+1} \times exp\left(\frac{1}{2d} \times \mathscr{G}_1\right) \\ i_j &\leftarrow x_n^{((j-1) \ mod \ \mu)+1} + \sigma_j \mathscr{G}_d \end{aligned}$$

    $\forall j \in \{1, \ldots, \lambda\}$, evaluate i_j Y times. Let y_j be the averaging over these Y evaluations.
    Define $j_1, \ldots, j_\lambda$ so that $y_{j_1} \leq y_{j_2} \leq \cdots \leq y_{j_\lambda}$.
    Compute σ_{n+1}^k and x_{n+1}^k for $k \in \{1, \ldots, \mu\}$:

$$\begin{aligned} \sigma_{n+1}^k &\leftarrow \sigma_{j_k} \\ x_{n+1}^k &\leftarrow x_{j_k} \end{aligned}$$

    $n \leftarrow n + 1$
end while
    return x_n^1

Chapter 4

Convergence rate of Evolution Strategies with additive noise: a lower bound

This chapter is based on:

Astete-Morales, S., Cauwet, M.-L., and Teytaud, O. (2015b). Evolution strategies with additive noise: A convergence rate lower bound. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, FOGA '15, pages 76–84, New York, NY, USA. ACM

Chapter 2 presented some lower bounds for zero-order value-based algorithms. It was stated that in the case of additive noise, the Simple Regret (SR) converges at best logarithmically with a slope asymptotically closed to -1. The logarithmic type of convergence of the Simple Regret has also been shown for Evolution Strategies in the presence of additive noise, however, no bound - upper or lower - has been shown on the rate of convergence. We address this question in the present chapter, where we consider the *SR* in expectation.

We recall the definition of the model of noise under study:

$$\forall\, x \in \mathscr{D} \subset \mathbb{R}^d,\ f(x,\omega) = f(x) + \omega. \tag{2.4}$$

We assume that $\mathbb{E}(\omega) = 0$ and $Var(\omega) = 1$.

For any integer n, x_n is the nth search point, y_n is its noisy evaluation and $\tilde{x}_n$ is the $(n-1)^{th}$ recommendation. The sequence of search points and their noisy evaluation is:

$$Z_n = ((x_0, y_0), \ldots, (x_{n-1}, y_{n-1})) \tag{4.1}$$

We say that the slope $-\alpha$ is *verified* on a family F of noisy objective functions if α is such that:

$$\forall f \in F,\ \exists C > 0,\ \forall n \in \mathbb{N},\ \mathbb{E}SR_n \leq C/n^{\alpha}. \tag{4.2}$$

This means that several different slopes might be verified. Note that the important is the supremum of such α.

In the following the inner product in $\mathbb{R}^d$ is represented by $\langle \cdot, \cdot \rangle$ and $sgn()$ refers to the sign function.

4.1 Theoretical analysis

This section consists of the formalization of the algorithms and the main result of the chapter. We present on one side the formalization of the concept of a *black-box noisy optimization algorithm* (Algorithm 4.1, Section 4.1.1) and on the other side a formalization of "classical" ES (Def. 4.1.1, Section 4.1.1). Notably, the ESs covered by the formalization of ES in Definition 4.1.1 correspond to a wide family but not all of them. The condition on Eq. 4.5 refers to the evolution of the step-size and the approximation to the optimum. The latter condition holds provably for some ESs, and probably for much more, but not necessarily for e.g. ESs with surrogate models or ESs with large mutations [Ong et al., 2003, Zhou et al., 2004, Beyer, 1998]. Also algorithms using several Gaussian distributions simultaneously or multimodal random variables for generating individuals are not covered. Thus, we mainly consider here ESs with one Gaussian distribution which scales roughly as the distance to the optimum.

In Section 4.1.2 we prove the theorem that states that the family of Evolution Strategies described by the formalization converges at best with rate $-1/2$ (tightness comes from [Coulom et al., 2011]).

4.1.1 Formalization of algorithms

General optimization framework

Basically, an optimization algorithm samples some search points, evaluates them and proposes a recommendation (i.e. an approximation of the optimum) from this information. We formalize a general optimization algorithm in Algorithm 4.1.

Algorithm 4.1 General optimization framework.

Input:

random seed s; parameter p; initial internal state $\mathscr{I}$; objective function f.

Output:

an approximation of the optimum of the objective function.

1: $n \leftarrow 0$
2: **while** not finished **do**
3: $\quad r \leftarrow rand(s)$
4: $\quad \tilde{x}_n \leftarrow \mathtt{R}(Z_n, r, n, p, \mathscr{I})$. ▷ Recommend
5: $\quad x_n \leftarrow \mathtt{SP}(Z_n, r, n, p, \mathscr{I})$. ▷ Search
6: $\quad y_n \leftarrow f(x_n, \omega_n)$ ▷ Evaluate f in search point
7: $\quad n \leftarrow n+1$.
8: **end while**

The procedures R and SP correspond to the *Recommendation* and *Search Point* stages of the algorithm. R outputs a feasible point that stands as the approximate optimum of the respective iteration and SP generates new search points to be evaluated. $\mathscr{I}$ represents the internal state of the algorithm, possibly modified inside SP. The sequence Z_n is as defined in Eq. 4.1.

When $n = 0$, Z_n is void and the points x_0 and $\tilde{x}_0$ are initialized depending on the parameters of the algorithm and on the random seed s. Starting from $n = 1$, both the R and the SP functions return values depending on the results of previous iterations.

The presented framework is in fact very general. First, if we consider algorithms that make use of populations for the optimization process, this characteristic can be simulated in the framework even when apparently the population size in Algorithm 4.1 is always 1. For example, let us say we want to check if an algorithm that uses a population of size λ (λ-population-based) can fit the framework. Then, an iteration on the λ-population-based algorithm can be "split" into several iterations in the framework, so that the λ individuals can be generated by λ iterations of a population size 1, just by adapting the R and SP functions.

Second, thanks to r, randomized algorithms are included. We propose different algorithms which match this framework in Section 4.2[1].

The framework presented encodes black-box algorithms and therefore both

[1] In particular Algorithm 4.2 in Section 4.2.1 is presented as an explicit example of an algorithm written to fit the framework described by Algorithm 4.1.

Evolution Strategies and algorithms reaching fast rates as those presented in [Fabian, 1967, Shamir, 2013]. Note that there is no restriction regarding the distance between x_n and $\tilde{x}_n$. In particular, in the case of [Fabian, 1967, Shamir, 2013, Coulom, 2011] the search points and the recommendation points can be far from each other (it is even desirable). On the contrary, ESs have a search point procedure that dictates that x_n should not be very far from the $\tilde{x}_n$.

Perimeter covered by general optimization framework

We provide some observations to clarify the scope of Algorithm 4.1, and how it covers the usual definitions of black-box optimization algorithms.

In general, a black-box optimization algorithm uses the objective function as an oracle. Since we consider a black-box setting, there is no access to any internal characteristic of the objective function.

On the other side, a black-box optimization algorithm has a state that is either its initial state (in case we are at the first time step) or a function of its internal state and of the results of requests to the oracle.

And since the algorithm is an algorithm for optimization, it must provide an approximation of the optimum. Such an approximation is termed "recommendation". We here decide that the approximation of the optimum should not change between two calls to the objective (i.e. oracle) function.

Therefore, an optimization algorithm is a sequence of internal computations, which modify an internal state. This sequence is sometimes interrupted by a call to the oracle function, or by a change in the recommendation.

We can then rewrite the algorithm, hiding all internal transformations of the internal state $\mathscr{I}$ between two calls to the oracle in some `SP` function. The algorithm then evaluates the objective function at x_n (call to the oracle). Next, it proposes a new approximation of the optimum; this computation is encoded in `R`. We have specified that this does not modify $\mathscr{I}$; but the procedure `R` can be duplicated inside `SP`, which is allowed to modify the internal state, if necessary, so this is not a loss of generality. The random seed is available for all functions so that there is no limitation around randomization.

We have assumed that the algorithm never spends infinite computation times between two calls to the oracle, and does not stop. We can just decide that in such a case we report the same output for `R` and the same output for `SP`.

All the elements discussed in this section allow us to use the general optimization framework described in Algorithm 4.1 to represent many of black-box optimization algorithms

Simple Evolution Strategies definition

ESs are black-box optimization algorithms and they fit the framework in Algorithm 4.1. Nonetheless, one important feature that characterizes them is the way to generate search points. Normally, the sampling of new search points is made "around" the recommendation point of the previous generation. This means that the SP procedure is defined by:

$$\mathtt{SP}(\mathfrak{Z}_n) = \mathtt{R}(\mathfrak{Z}_n) + \sigma(\mathfrak{Z}_n)\Psi(\mathfrak{Z}_n) \tag{4.3}$$

where, for short, $\mathfrak{Z}_n = (Z_n, r, n, p, \mathscr{I})$. The step-size $\sigma(\mathfrak{Z}_n)$ is usually updated at each generation. $\Psi(\mathfrak{Z}_n)$ is an independent d-dimensional zero-mean random variable, not necessarily Gaussian, with

$$\mathbb{E}\|\Psi(\mathfrak{Z}_n)\|^2 = d. \tag{4.4}$$

Also, we consider that the ESs should satisfy the following condition on the evolution of the step-size with regards to the recommendation points. In the following section we will explain with more details the reasons behind this condition:

$$\exists D > 0,\ \forall n \geq 0,\ \mathbb{E}[\sigma(\mathfrak{Z}_n)^2] \leq D\mathbb{E}[\|\tilde{x}_n - x^*\|^2]. \tag{4.5}$$

Now we can state the definition of ES covered by the theorem in Section 4.1.2.

Definition 4.1.1. Simple Evolution Strategy. *A Simple Evolution Strategy is an algorithm that matches the framework of Alg. 4.1 and satisfies both Eq. 4.3 and Eq. 4.5.*

Perimeter covered by Simple Evolution Strategy definition

Let us discuss the assumptions in our Evolution Strategy framework above.

Eq. 4.4 is not a strong constraint, as one can always rephrase the algorithm for moving multiplicative factors from $\Psi(\mathfrak{Z}_n)$ to $\sigma(\mathfrak{Z}_n)$ so that $\mathbb{E}\|\Psi(\mathfrak{Z}_n)\|^2 = d$.

The assumption in Eq. 4.3 is easy to understand. It is verified for a classical ES with a single parent or a μ/μ recombination (i.e. parent equal to the average of selected offspring), including weighted recombinations.

The assumption in Eq. 4.5 is more difficult to grasp. It means that $\tilde{x}_n$ and $\sigma(\mathfrak{Z}_n)$ decrease at the same rate towards the optimum. The literature provides the following cases:

- The scale-invariant algorithm obviously verifies the assumption, by definition. The scale-invariant algorithm is however essentially a theoretical algorithm, used for theoretical proofs rather than for real applications.

- Related results are proved for some ESs in the noise-free case, as shown in [Auger, 2005]; $\sigma(\mathfrak{Z}_n)/\|\tilde{x}_n - x^*\|$ converges to some distribution. The work in [Astete-Morales et al., 2014] shows that it is also true for some provably convergent noisy optimization evolutionary algorithm with resamplings. However, it is not clear that these results imply Eq. 4.5.

- Beyond mathematical proofs (indeed there are many evolutionary algorithms for which we have no convergence proof at all), Eq. 4.5 is widely verified in experimental results when algorithms converge, in the $(1+1)$-ES [Rechenberg, 1973], in self-adaptive algorithms [Beyer and Schwefel, 2002], in Covariance Matrix Adaptations variants [Hansen and Ostermeier, 2001], and indeed most ESs [Beyer, 2001].

What would be an EA which does *not* verify Eq. 4.5? A natural example is an Evolutionary Algorithm which samples far from the current estimate $\tilde{x}_n$ of the optimum, e.g. for building a surrogate model. Interestingly, all optimization algorithms which are fast in noisy optimization with constant noise variance in the vicinity of the optimum verify such a property, namely sampling far from the optimum [Fabian, 1967, Shamir, 2013, Coulom, 2011]. This suggests that modified ESs which include samplings far from the optimum, might be faster.

4.1.2 Lower bound for Simple Evolution Strategies

We now state our main theorem, namely the proof that Evolution Strategies, in their usual setting without mutations far from the optimum, can not reach a rate as good as algorithms without such restrictions.

Theorem 4.1.1. *Let F be the set of quadratic functions $f : \mathscr{D} \to \mathbb{R}$ defined on $\mathscr{D} = \mathbb{R}^d$ by $f(x) = \frac{1}{2}\|x\|^2 - \langle x^*, x\rangle$ for some $\|x^*\| \leq \frac{1}{2}$. Consider a simple Evolution Strategy as in definition 4.1.1 and the noisy optimization of $f \in F$ corrupted by some additive noise with variance 1 in the unit ball: $f(x,\omega) = f(x) + \omega$ such that $\mathbb{E}(\omega) = 0$. Then, for all $\alpha > \frac{1}{2}$, the slope $-\alpha$ is not reached.*

Remark 4.1.1. (Tightness in the framework of Evolution Strategies.) *The work in [Coulom et al., 2011] shows that, within logarithmic factors, an Evolution Strategy with Bernstein races (with modified sampling in order to avoid huge*

numbers of resamplings due to individuals with almost equal fitness values) can reach a slope $-\alpha$ arbitrarily close to $-\alpha = -\frac{1}{2}$. To the best of our knowledge, it is not known whether we can reach $\alpha = \frac{1}{2}$.

Remark 4.1.2. (Scope of the lower bound) *Note that Theorem 4.1.1 considers a particular set of quadratic functions, but the result is valid for any family of functions that includes sphere functions.*

Proof. Let us assume, in order to get a contradiction, that a slope $\alpha > \frac{1}{2}$ is reached. Then, $\mathbb{E}SR_n \leq C/n^\alpha$ for some $\alpha > 1/2$ and $C > 0$.
Notations are similar to Section 4.1.1: for any $i \in \{1,\dots,n\}$, $x_i = \mathtt{SP}(\mathfrak{Z}_i)$, $\tilde{x}_i = \mathtt{R}(\mathfrak{Z}_i)$, $\sigma_i = \sigma(\mathfrak{Z}_i)$, $\Psi_i = \Psi(\mathfrak{Z}_i)$, where Ψ_i are centered independent random variables in $\mathbb{R}^d$ with $\mathbb{E}\|\Psi_i\|^2 = d$. Let us evaluate the Cumulative Regret:

$$\begin{aligned}
2\mathbb{E}CR_n &= 2\sum_{i=1}^{n}\left(\mathbb{E}f(x_i,\omega_i) - f(x^*)\right) \text{by definition in Eq. 2.11.} \\
&= \sum_{i=1}^{n}\mathbb{E}[\|x_i\|^2 - 2\langle x^*, x_i\rangle + \|x^*\|^2] \\
&= \sum_{i=1}^{n}\mathbb{E}[\|x_i - x^*\|^2] \\
&= \sum_{i=1}^{n}\mathbb{E}[\|\tilde{x}_i - x^* + \sigma_i\Psi_i\|^2] \text{ by Eq. 4.3} \\
&\leq \sum_{i=1}^{n}\left(\mathbb{E}\|\tilde{x}_i - x^*\|^2 + \mathbb{E}\sigma_i^2\mathbb{E}\Psi_i^2\right) \text{ by independence} \\
&\leq \sum_{i=1}^{n}\left(\mathbb{E}\|\tilde{x}_i - x^*\|^2 + d\mathbb{E}\sigma_i^2\right) \text{ by Eq. 4.4} \\
&\leq 2(1+dD)\sum_{i=1}^{n}\mathbb{E}[SR_i] \text{ by Eq. 4.5}
\end{aligned}$$

The last equation leads to

$$\mathbb{E}CR_n \leq C(1+dD)n^{1-\alpha} \tag{4.6}$$

[Shamir, 2013, Theorem 6] has shown that, for any optimization algorithm as defined in Section 4.1.1, there is at least one function in $f \in F$ for which the Cumulative Regret is $CR_n \geq 0.02\min(1, d\sqrt{n})$, which contradicts Eq. 4.6. □

4.2 Experimental verification of the lower bound

This section is devoted to the verification of the lower bound on the convergence rate for ESs stated in Theorem 4.1.1 and the comparison with the convergence rate of a "fast" algorithm: Shamir's algorithm [Shamir, 2013].

We show experimentally that the rate -1 promised by the results in [Shamir, 2013] is visible on experiments, even with moderate budgets in terms of numbers of function evaluations. We then show that, consistently with theory, we could not do better than slope $-1/2$ with ESs (Section 4.2.2). The experimental results presented on this section use an approximation of the slope of Simple Regret (Eq. 3.1)[2]:

$$\log(\mathbb{E}SR_n/d)/\log(n)$$

4.2.1 Fast convergence: Shamir's algorithm

[Shamir, 2013] designed an optimization algorithm using stochastic approximation methods ([Polyak and Tsybakov, 1990, Spall, 2000]). At each iteration, it computes an approximate gradient and uses it to update the estimate of the optimum. This algorithm is described in Algorithm 4.2 and is named Shamir's algorithm in the following. Note that the search points defined in the procedure `SP` can be far from the current approximation thanks to the random direction r. This algorithm provably asymptotically reaches some slope arbitrarily close to $\alpha = 1$ in the quadratic case. Importantly, we present the algorithm in the framework of Section 4.1.1, however, neither Eq. 4.3, nor Eq. 4.5 are satisfied so that this cannot be considered a Simple Evolution Strategy as in Def. 4.1.1.

[2] Note that dividing by d does not matter asymptotically and both theory [Shamir, 2013] and experiments show that it is a good normalization for convergence rates.

Algorithm 4.2 Shamir's algorithm, written in the general optimization framework.

procedure $\mathtt{R}(x_0,\dots,x_{n-1},y_0,\dots,y_{n-1},r,n,p=(\lambda,\varepsilon,B,d),\mathscr{I})$
$\tilde{x}_n \leftarrow \frac{2}{n}\sum_{j=\lceil n/2\rceil,\dots,n}^{n}\mathscr{I}_j$ ▷ $\mathscr{I}$ is a vector of n elements in the domain.
  return $\tilde{x}_n$
end procedure
procedure $\mathtt{SP}(x_0,\dots,x_{n-1},y_0,\dots,y_{n-1},r,n,p=(\lambda,\varepsilon,B,d),\mathscr{I})$
  if $n=0$ **then**
    $\mathscr{I}\leftarrow(0)$
    return $x_0=0$
  end if
$x_n \leftarrow x_{n-1}+\frac{\varepsilon}{\sqrt{d}}r$
$\tilde{g}\leftarrow\frac{\sqrt{d}y_{n-1}}{\varepsilon}r$
  if $\|x_{n-1}-\frac{1}{\lambda n}\tilde{g}\|\geq B$ **then**
    $x_{n-1}-\frac{1}{\lambda n}\tilde{g}\leftarrow B\frac{x_{n-1}-\frac{1}{\lambda n}\tilde{g}}{\|x_{n-1}-\frac{1}{\lambda n}\tilde{g}\|}$
  end if
$\mathscr{I}\leftarrow(\mathscr{I},x_{n-1}-\frac{1}{\lambda n}\tilde{g})$
  return x_n
end procedure

Input:
$p=(\lambda,\varepsilon,B,d)\in\mathbb{R}_+\times(0,1]\times\mathbb{R}_+\times\mathbb{N}^*$, $s=$ *random seed*, objective function f.
Output:
approximation of the optimum x^* of the objective function f

$n\leftarrow 0$
while not finished **do**
  Generate $r\in\{-1,1\}^d$, uniformly and randomly
  $\tilde{x}_n\leftarrow\mathtt{R}(x_0,\dots,x_{n-1},y_0,\dots,y_{n-1},r,n,p,\mathscr{I})$
  $x_n\leftarrow\mathtt{SP}(x_0,\dots,x_{n-1},y_0,\dots,y_{n-1},r,n,p,\mathscr{I})$
  $y_n\leftarrow f(x_n,\omega)$
  $n\leftarrow n+1$
end while
  return $\tilde{x}_n$

We compare the performance of Shamir's algorithm on the noisy sphere func-

tion: $x \mapsto \|x-0.5\|^2 + \mathcal{G}(0,1)$, where $\mathcal{G}(0,1)$ is a standard Gaussian.

Results are presented in Figure 4.1. Experiments are performed in various dimensions: 2, 4, 8 and 16. We observe that independently of the dimension, the algorithm's slope is clearly smaller than $-1/2$ (i.e. faster than the bound we have proved for the Simple Evolution Strategies).

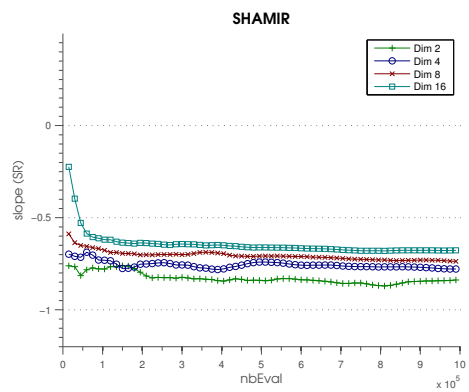


Figure 4.1: Shamir's algorithm [Shamir, 2013] on the sphere function $x \mapsto \|x\|^2 + \mathcal{G}(0,1)$ where $\mathcal{G}(0,1)$ is an independent Gaussian standard random variable. X-axis: number of function evaluations. Y-axis: estimate of the slope (see Eq. 4.2). The maximum standard deviation for all averages presented here (experiments are averaged over 21 runs) is 10^{-3}.

4.2.2 Slow convergence: UH-CMA-ES and (1+1)-ES

We here experiment with the UH-CMA-ES algorithm introduced in Chapter 2 on $f(x) = \|x-x^*\|^2 + 0.3\mathcal{G}(0,1)$, where $\mathcal{G}(0,1)$ is a standard Gaussian random variable and with $x^* = 0.5$. In these experiments, we use all the default parametrizations of UH-CMA-ES[3]. Results are shown in Figure 4.2. Even though the rate of convergence to the optimum decreases as the dimension increases, we can observe rates of between -0.1 and -0.3, all of them greater than -0.5.

Now let us consider a Simple Evolution Strategy, namely the $(1+1)$ES with one-fifth rule [Rechenberg, 1973, Schwefel, 1974], with additional revaluations, implemented as shown in Algorithm 4.3.

[3] See settings at URL `https://www.lri.fr/~hansen/cmaes.m`.

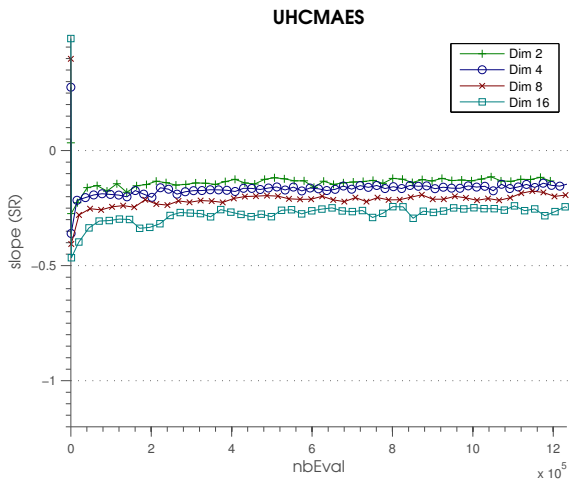


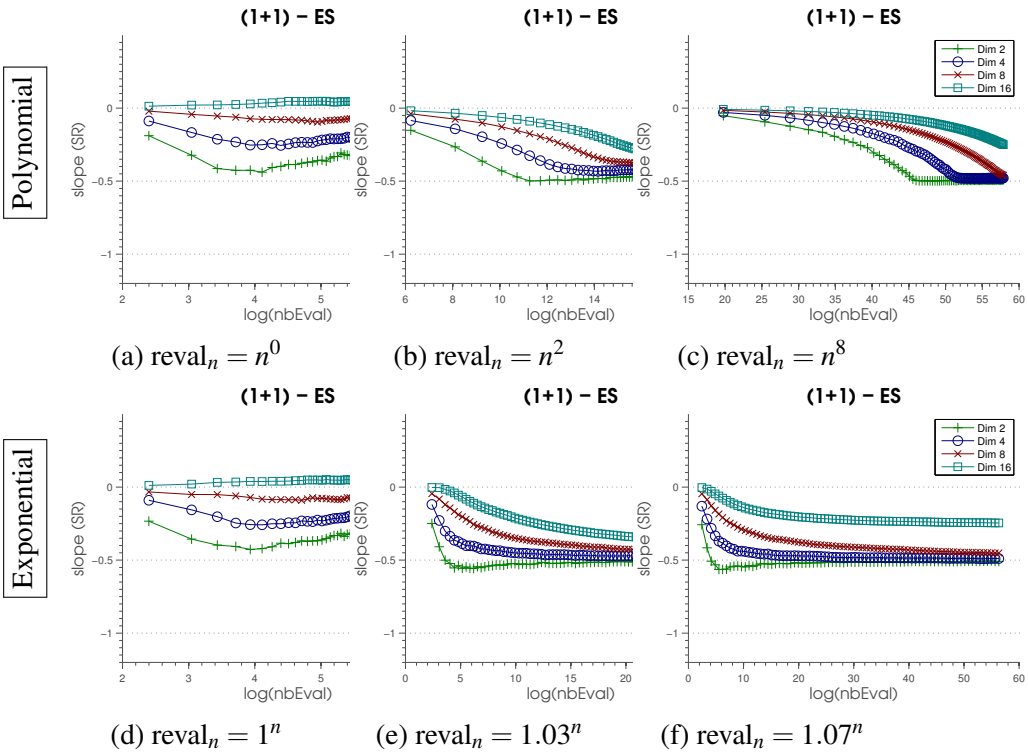
Figure 4.2: UH-CMA-ES algorithm [Hansen et al., 2009] on the sphere function $x \mapsto \|x\|^2 + 0.3\mathscr{G}(0,1)$ where $\mathscr{G}(0,1)$ is an independent Gaussian standard random variable. The maximum standard deviation for all averages presented here (experiments are averaged over 21 runs) is 1.

A way to slightly improve Algorithm 4.3 is to improve the computation of the fitness value of the current best recommendation by averaging the current estimate with the previous estimates of the same search point, when the mutation has not been accepted and $x_n = x_{n-1}$. We propose such a modification in Algorithm 4.4 by using a weighted average in the estimate fitness value of the current best search point.

Experiments on the noisy sphere function $\|x - x^*\|^2 + \mathscr{G}(0,1)$, where $\mathscr{G}(0,1)$ is a standard Gaussian, are provided, using Algorithm 4.4. Results are presented in Figure 4.3 in various dimensions (2, 4, 8, 16 respectively). Seemingly both exponential and polynomial resamplings lead to a slope $-1/2$.

4.3 Conclusion

We have shown that Evolution Strategies, at least in their most common form, can not reach the same rate as noisy optimization algorithms which use evaluations of the objective function farther from the approximate optimum in order to obtain extra information on the function. On the contrary, ESs use evaluations of objective functions only in the neighborhood of the optimum. Therefore, usual ESs cannot



(a) $\text{reval}_n = n^0$ (b) $\text{reval}_n = n^2$ (c) $\text{reval}_n = n^8$

(d) $\text{reval}_n = 1^n$ (e) $\text{reval}_n = 1.03^n$ (f) $\text{reval}_n = 1.07^n$

Figure 4.3: Results $(1+1)$-ES for dimension 2, 4, 8 and 16. First row of plots presents Polynomial resampling and the second row Exponential resampling. The maximum standard deviation for all averages presented here (experiments are averaged over 400 runs) is 0.025. Note that (nbEval) is the number of evaluations, n.

Algorithm 4.3 $(1+1)-ES$ for noisy optimization with resamplings. $\mathscr{G}(0,1)$ is a standard Gaussian. The function 'number_of_revaluations' depends on the current iteration and on a parameter p. Typically the number of revaluations is polynomial: n^p or exponential: p^n.

Input:
Initial search point $x = (0,\ldots,0)$; initial step-size $\sigma = 1$; coefficient p; function 'number_of_revaluations'; objective function f.

Output:
an approximation of the optimum of the objective function.

```
n ← 0
while not finished do
    x' ← x + σ𝒢(0,1)
    n ← n + 1
    r ← number_of_revaluations(n, p)
    y ← (1/r) Σ_{i=1}^{r} f(x, ω_i)
    y' ← (1/r) Σ_{i=1}^{r} f(x', ω_i)
    if y' < y then
        x ← x' and σ ← 2σ
    else
        σ ← .84σ
    end if
end while
    return x
```

reach rates as the ones in [Fabian, 1967, Shamir, 2013] (Shamir [Shamir, 2013] in the quadratic case non-asymptotically, Fabian [Fabian, 1967] in the general case but asymptotically). The latter type of algorithms reach a slope -1, whereas we have a limit at $-1/2$ with Evolution Strategies. This also shows the optimality of the rate $-1/2$ obtained by Racing-based Estimation of Distribution Algorithms (R-EDA) [Rolet and Teytaud, 2010a], in the framework of local sampling only.

It is important to note that the result in this chapter indeed covers not only Evolutionary Algorithms, but also, for example, many pattern search methods. We proved the results for algorithms which perform sampling within some distance of the approximate optimum, this distance scaling roughly as the distance to the optimum. This property is satisfied by a large family of ESs (see Section 4.1.1). However, for many algorithms it is verified only experimentally and not formally

proved.

ESs with surrogate models are not concerned by our lower bound. More precisely, if we include strong surrogate modelling with large mutations (and so contradicting Eq. 4.5), then we can recover fast rates with slope -1. An extreme example of this situation is the case in which the sampling/surrogate model is exactly the algorithm in [Fabian, 1967], [Shamir, 2013] or [Coulom, 2011]. Using them as to obtain surrogate models within an ES will ensure a fast convergence rate for the ES. Obviously, it is desirable to verify if such result can also be obtained with more "evolutionary" approaches.

The bound presented in this chapter does not cover evolutionary algorithms that would use very large mutations [Beyer, 1998]. Maybe this is a good path to follow for designing fast evolutionary noisy optimization algorithms.

For all experiments we check convergence rates on the sphere function with additive noise.
We consider an algorithm with theoretical fast rate, the Shamir's algorithm, and two ESs: UH-CMA-ES and $(1+1)$-ES. For Shamir's algorithm we have achieved a successful implementation of the algorithm [4] and confirmed empirically the fast convergence rate proved in [Fabian, 1967, Shamir, 2013] (i.e. slope of $SR = -1$). For UH-CMA-ES and $(1+1)$-ES we have shown that ESs can approximate slope of SR -0.5 using $(1+1)$-ES. UH-CMA-ES also reaches linear convergence in the log-log scale but with a slower rate (slope of SR around -0.2).

Further work. A first further work consists in proving the result in a wider setting, this is, weakening the assumption in Eq. 4.5. We might also check other criteria than non-asymptotic expected Simple Regret, e.g. almost sure convergence. Another further work is investigating which optimization algorithms, other than Evolution Strategies, are concerned by our result or by similar results. In the case of strongly convex functions with a lower bound on eigenvalues of the Hessian, we conjecture that the asymptotic rate -1 can also not be reached by the considered family of evolutionary algorithms.

[4] [Shamir, 2013] delivers only the theoretical analysis of his algorithm.

Algorithm 4.4 Slightly improved $(1+1)-ES$ for noisy optimization with resamplings.

Input:

Initial search point $x = (0,\ldots,0)$; initial step-size $\sigma = 1$; coefficient p; function 'number_of_revaluations'; objective function f.

Output:

an approximation of the optimum of the objective function.

```
n ← 0
k ← 0
y ← 0
while not finished do
    x' ← x + σ𝒢
    n ← n + 1
    r ← number_of_revaluations(n, p)
    y ← k × y + r × (1/r) Σ_{i=1}^{r} f(x, ω_i)
    k ← k + r
    y ← y/k
    y' ← (1/r) Σ_{i=1}^{r} f(x', ω_i)
    if y' < y then
        x ← x'
        σ ← 2σ
        y ← 0
        k ← 0
    else
        σ ← .84σ
    end if
end while
```

Chapter 5

Comparison-based algorithms can be fast!

This chapter is based on:

Cauwet, M.-L. and Teytaud, O. (2016b). Noisy optimization: Fast convergence rates with comparison-based algorithms. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016*, pages 1101–1106

In the previous chapter, we have shown that a wide range of comparison-based algorithms without large mutations have a Simple Regret at best in $O(1/\sqrt{n})$ in expectation. Stochastic gradient descent can reach (tightly) a Simple Regret in $O(1/n)$ (see Chapter 2). It has been conjectured in [Shamir, 2013] that gradient approximation by finite differences (hence, not a comparison-based method) is necessary for reaching such a $O(1/n)$. We answer this conjecture in the negative, providing a comparison-based algorithm as good as gradient methods, i.e. reaching $O(1/n)$ - under the condition, however, that the noise is Gaussian. Experimental results confirm the $O(1/n)$ Simple Regret, i.e., squared rate compared to many published results at $O(1/\sqrt{n})$.

In this chapter, we assume that the objective function is perturbed by a centred Gaussian noise:

$$\forall x \in \mathscr{D},\ f(x,w) = f(x) + \mathscr{G}(0,b),$$

where $\mathscr{G}(a,b)$ is a Gaussian random variable with mean a and standard deviation $b > 0$. As usual x^* denotes the minimum of the noise-free function f.

Section 5.1 describes the key idea to get a fast comparison-based algorithm in a noisy setting. The theoretical aspects and a precise description of a fast optimization algorithm is given in Section 5.2 for the specific case of the sphere function. In this case, the technicality in the proof is lighter and allowed a good insight of what we will use when switching to a larger family of functions: the quadratic functions in Section 5.3. Last, we address the experimental aspects in Section 5.4.

5.1 Comparison procedure

The main idea is to estimate the parameters of the objective function. The algorithm hence builds a model of the function and provides an approximation of the optimum. Specifically, comparing 2 search points n times provides an estimation at distance $O(1/\sqrt{n})$ of one parameter of the function. This estimation is made possible through the frequency at which the fitness values of one of the search points is better than the other. In particular, it is crucial to know the model of noise. Hence, the optimization algorithms of Sections 5.2 and 5.3 consist in a sequence of calls to COP, given below.

Comparison procedure (COP).

```
procedure COP(n, x, y, f(·, ω))
    for i = 1 to n do
        f_x^i ← f(x, ω_{2i})
        f_y^i ← f(y, ω_{2i+1})
    end for
    F ← (1/n^2) Σ_{1≤i,j≤n} 1_{f_x^i < f_y^j}
    return F
end procedure
```

Importantly, this operator can be computed faster than the apparent $O(n^2)$ complexity. Using sorting algorithm, the complexity is $O(n \log n)$. [Jamieson et al., 2012] have presented a bound for a comparison-based operator, using a number of comparisons quadratic $O\left(\frac{1}{\varepsilon^2}\right)$ for ensuring precision ε in the Simple Regret - whereas we only need $O\left(\frac{1}{\varepsilon}\log\frac{1}{\varepsilon}\right)$ comparisons.

5.2 Sphere function

5.2.1 In dimension 1

We first propose in Alg. 5.1 an algorithm (COPS1) achieving regret $O(1/n)$ on the noisy sphere problem in dimension 1.

Algorithm 5.1 Comparison procedure for sphere function in dimension 1 (COPS1).

Input:
oracle $f(\cdot,\omega) : x \in \mathbb{R} \mapsto \mathcal{G}(|x-x^*|^2, 1)$; even budget n.

Output:
approximation $\tilde{x}$ of the optimum $x^* \in [-1,1]$ of the objective function $f:\ x \mapsto |x-x^*|^2$

1: $K \leftarrow n/2$
2: $F \leftarrow \text{COP}(K, 1, -1, f(\cdot,\omega))$
3: Define $\tilde{x}$ such that $\mathbb{P}\left(\mathcal{G}(0,1) < \sqrt{8}\tilde{x}\right) = F$
4: $\tilde{x} \leftarrow \max(-1, \min(1, \tilde{x}))$
return $\tilde{x}$

Theorem 5.2.1. *Let $f(x,\omega) = |x-x^*|^2 + \mathcal{G}(0,1)$ be the noisy sphere function in dimension 1, where $x^* \in [-1,1]$. Then the Simple Regret of* COPS1 *after n evaluations satisfies:*

$$\mathbb{E}SR_n = O(1/n). \tag{5.1}$$

Proof. Consider COPS1 on such an objective function. By definition of $f(\cdot,\omega)$ and f,

$$\begin{aligned} p &= \mathbb{P}(f(1,\omega) < f(-1,\omega)) \\ &= \mathbb{P}\left(|1-x^*|^2 + \mathcal{G}(0,1) < |-1-x^*|^2 + \mathcal{G}(0,1)\right) \\ &= \mathbb{P}\left(\sqrt{2}\mathcal{G}(0,1) < (1+x^*)^2 - (1-x^*)^2\right) \\ &= \mathbb{P}(\mathcal{G}(0,1) < \sqrt{8}x^*). \end{aligned} \tag{5.2}$$

Step 1: Expectation and Variance of F.
With the notations of COP, let us define:

$$\forall\, i,j \in \{1,\dots,n\}^2,\ \mathbf{1}_{i,j} = \begin{cases} 1 \text{ if } f_1^i < f_{-1}^j \\ 0 \text{ otherwise} \end{cases}$$

$\mathbf{1}_{i,j}$ is Bernoulli distributed with probability of success p.
F is the output of the COP procedure. By definition,

$$F = \frac{1}{K^2} \sum_{1 \le i,j \le K} \mathbf{1}_{i,j}.$$

The expectation and variance of F are then:

$$\begin{aligned} \mathbb{E}F &= p \\ Var(F) &= \frac{1}{K^4} \sum_{i=1}^{K} \sum_{j=1}^{K} Cov\left(\sum_{k=1}^{K} \mathbf{1}_{i,k}, \sum_{k'=1}^{K} \mathbf{1}_{j,k'} \right) \\ &= \frac{1}{K^4} \sum_{i=1}^{K} \sum_{j=1}^{K} \sum_{k=1}^{K} \sum_{k'=1}^{K} Cov(\mathbf{1}_{i,k}, \mathbf{1}_{j,k'}) \end{aligned} \tag{5.3}$$

If $i \neq j$ and $k \neq k'$, $Cov(\mathbf{1}_{i,k}, \mathbf{1}_{j,k'}) = 0$ by independence. If $i = j$ (or $k = k'$), by Cauchy-Schwarz:

$$Cov(\mathbf{1}_{i,k}, \mathbf{1}_{i,k'}) \le \sqrt{Var(\mathbf{1}_{i,k}) Var(\mathbf{1}_{i,k'})} \le \frac{1}{4}$$

This together with Eq. 5.3 give:

$$\begin{aligned} Var(F) &= \frac{1}{K^4} \left(\sum_{i=1}^{K} \sum_{k=1}^{K} \sum_{k'=1}^{K} Cov(\mathbf{1}_{i,k}, \mathbf{1}_{i,k'}) + \right. \\ &\qquad \left. \sum_{i=1}^{K} \sum_{j=1}^{K} \sum_{k=1}^{K} Cov(\mathbf{1}_{i,k}, \mathbf{1}_{j,k}) \right), \\ &\le \frac{1}{K^4} \times \frac{K^3}{2}, \\ &\le \frac{1}{n}. \end{aligned}$$

Step 2: Lipschitz. We denote by Φ the cumulative distribution function of the standard Gaussian: $\Phi(x) = \mathbb{P}(\mathscr{G}(0,1) < x)$ and m and M such that $\Phi^{-1}_{[m,M]} : [m,M] \to [-1,1]$ is the inverse of Φ over these intervals. Let us define

$$h(x) = \begin{cases} \Phi^{-1}_{[m,M]}(x) & \text{if } m \leq x \leq M \\ -1 & \text{if } x < m \\ 1 & \text{if } M < x \end{cases}$$

Let us evaluate the Lipschitz coefficient $L(h)$ of h. $\Phi^{-1}_{[m,M]}$ is differentiable over $[m,M]$ since Φ is differentiable over $[-1,1]$ hence its Lipschitz $L(\Phi^{-1}_{[m,M]})$ is bounded. h is continuous, and h is constant over $(-\infty,m]$ and $[M,\infty)$; hence the Lipschitz of h is $L(\Phi^{-1}_{[m,M]})$ over $[m,M]$.

Step 3: Concluding. We have, by definition of COPS1 for $\tilde{x}$ and by Eq. 5.2 for x^*,

$$\tilde{x} = \frac{h(F)}{\sqrt{8}} \text{ and } x^* = \frac{h(p)}{\sqrt{8}}, \tag{5.4}$$

By definition of the Simple Regret in Eq. 2.9,

$$\begin{aligned} \mathbb{E}SR_n &= \mathbb{E}|\tilde{x} - x^*|^2 \\ &\leq \mathbb{E}L(h)^2|F-p|^2/8 \text{ by } \textbf{Step 2} \\ &\leq \frac{L(h)^2}{8n} \text{ by } \textbf{Step 1}. \end{aligned}$$

□

Remark 5.2.1. *The result of Theorem 5.2.1 is based on the fact that the noise is a standard Gaussian. However, this result still holds as soon as the noise distribution has expectation 0, finite variance (possibly unknown, see Section 5.3) and a bounded Lipschitz. The distribution of the noise, on the other hand, must be known.*

5.2.2 Multidimensional sphere function

Alg. 5.2 (COPS) presents a straightforward extension to the noisy multidimensional sphere. $B_d(c,r)$ denotes the ball of center c and radius r in dimension d, and $\|.\|$ is the Euclidean norm.

Algorithm 5.2 Comparison procedure for the sphere function (COPS).

Input:
oracle $f(\cdot,\omega) : x \in \mathbb{R}^d \mapsto \mathcal{G}(\|x-x^*\|^2,1)$; budget n (multiple of $2d$).

Output:
approximation $\tilde{x}$ of the optimum $x^* \in B_d(0,1)$ of the objective function $f:\ x \mapsto \|x-x^*\|^2$

1: $K \leftarrow n/2d$
2: **for** $i = 1$ to d **do**
3: Apply COPS1 with a budget K on the unidimensional restriction of $f(\cdot,\omega)$ to $\{0\}^{i-1} \times [-1,1] \times \{0\}^{d-i}$
4: $\tilde{x}_i$ be the obtained approximation of the optimum in $[-1,1]$.
5: **end for**
return $\tilde{x} = (\tilde{x}_1,\ldots,\tilde{x}_d)$.

Theorem 5.2.2. *Let $f(x,\omega) = \|x-x^*\|^2 + \mathcal{G}(0,1)$ be the noisy sphere function, with $x^* \in B_d(0,1) \subset \mathbb{R}^d$. Then the Simple Regret of* COPS *after n evaluations is:*

$$\mathbb{E}SR_n = O(d/n).$$

Proof. The conditions of Theorem 5.2.1 are verified for each application of COPS1. The Simple Regret for the multidimensional case is the sum of the Simple Regrets of each restrictions. □

5.3 General quadratic forms

Alg. 5.3 extends the principle of Section 5.2 to the optimization of a wider class of quadratic functions. $\|\cdot\|_2$ denotes the matrix norm induced by $\|\cdot\|$, i.e $\|A\|_2 = \sup_{x\neq 0} \frac{\|Ax\|}{\|x\|}$ and $\|\cdot\|_F$ is the Frobenius norm. (e_i) is the standard basis and A^t is the transpose of matrix A.

Theorem 5.3.1. *Let $\varepsilon \in]0,1[$. Consider an objective function $f(x,\omega) = x^t Ax + Bx + C + D\mathcal{G}(0,1)$, with optimum x^* in $B_d(0,1-\varepsilon) \subset \mathbb{R}^d$, and $D > 0$. Assume that $\frac{1}{D}\|B\| \leq 1$ and $\frac{1}{D}|C| \leq 1$. If A is symmetric positive definite such that its eigenvalues are lower bounded by some $c > 0$ and $\|\frac{1}{D}A\|_2 \leq 1$, then, when applying* COPQUAD, $\mathbb{E}SR_n = O(\max((\lambda_{\max}/\lambda_{\min})^2, \lambda_{max}^2)D^2/n)$,

Algorithm 5.3 Comparison procedure for quadratic functions (COPQUAD).

Input:

oracle $f(\cdot,\omega) : x \in \mathbb{R}^d \mapsto \mathscr{G}(x^tAx+Bx+C,D)$; budget n (multiple of $d(d+3)-2$).

Output:

approximation $\tilde{x}$ of the optimum $x^* \in B_d(0,1)$ of the objective function $f:\ x \mapsto x^tAx+Bx+C$

$K \leftarrow \frac{n}{d(d+3)-2}$
for $i = 1$ to d **do**
  $F_{-e_i,e_i} \leftarrow \text{COP}(K,-e_i,e_i,f(\cdot,\omega))$
  Define $\hat{B}_i(D)$ such that $\mathbb{P}\left(\mathscr{G}(0,1) < \sqrt{2}\hat{B}_i(D)\right) = F_{-e_i,e_i}$
  $\hat{B}_i(D) \leftarrow \max(-5,\min(\hat{B}_i(D),5))$ ▷ Estimate of B_i/D

  $F_{0,e_i} \leftarrow \text{COP}(K,0,e_i,f(\cdot,\omega))$
  Define $\theta_{ii}(D)$ such that $\mathbb{P}\left(\mathscr{G}(0,1) < \theta_{ii}(D)/\sqrt{2}\right) = F_{0,e_i}$
  $\theta_{ii}(D) \leftarrow \max(-5,\min(\theta_{ii}(D),5))$
  $\hat{A}_{i,i}(D) \leftarrow \theta_{ii}(D) - \hat{B}_i(D)$ ▷ Estimate of $A_{i,i}/D$
end for
for $i = 1$ to d **do**
  for $j = i+1$ to d **do**
    $F_{0,e_i+e_j} \leftarrow \text{COP}(K,0,e_i+e_j,f(\cdot,\omega))$
    Define $\theta_{ij}(D)$ such that

$$\mathbb{P}(\mathscr{G}(0,1) < \theta_{ij}(D)/\sqrt{2}) = F_{0,e_i+e_j}$$

    $\theta_{ij}(D) \leftarrow \max(-5,\min(\theta_{ij}(D),5))$
    $\hat{A}_{i,j}(D) \leftarrow \frac{1}{2}(\theta_{ij}(D) - \hat{B}_i(D)$
      $-\hat{A}_{i,i}(D) - \hat{B}_j(D) - \hat{A}_{j,j}(D))$
    $\hat{A}_{j,i}(D) \leftarrow \hat{A}_{i,j}(D)$ ▷ Estimate of $A_{i,j}/D$ and $A_{j,i}/D$
  end for
end for
$\hat{A}(D) \leftarrow (\hat{A}_{i,j}(D))$
$\hat{B}(D) \leftarrow (\hat{B}_i(D))$
if $\hat{A}(D)$ is not singular **then**
  $\tilde{x} \leftarrow -\frac{1}{2}\hat{B}(D)^t\hat{A}(D)^{-1}$
else
  $\tilde{x} \leftarrow 0$
end if
  return $\tilde{x} \leftarrow$ projection of $\tilde{x}$ on $B_d(0,1)$.

---

where $\lambda_{\max}$ is the maximum eigenvalue of $\frac{1}{D}A$, and $\lambda_{min} > \frac{1}{D}c$ is the minimum eigenvalue.

Proof. Let x and y be two points to be compared in COPQUAD: $(x,y) \in \mathscr{C} := \{(e_i,-e_i)_i,(0,e_i)_i,(0,e_i+e_j)_{i\neq j}\}$. We denote by $\Delta_{x,y}$ the value $\Delta_{x,y} := \mathbb{E}(f(y,\omega)-f(x,\omega)) = f(y)-f(x)$ and by $F_{x,y}$ the frequency $F_{x,y} := \frac{1}{K^2}\sum_{1\leq i,j\leq K}\mathbf{1}_{f_x^i<f_y^j}$, where f_x^i and f_y^j are as in Section 5.1.

Step 1: Mean Squared Error of frequencies.

Similarly to step 2 of Theorem 5.2.1, and using the notation $\Phi(x) = \mathbb{P}(\mathscr{G}(0,1) < x)$,

$$\mathbb{E}(F_{x,y}) = \Phi\left(\frac{\Delta_{x,y}}{\sqrt{2}D}\right)$$
$$\mathbb{E}\left(F_{x,y} - \Phi\left(\frac{\Delta_{x,y}}{\sqrt{2}D}\right)\right)^2 = Var(F_{x,y}) = O(1/n). \tag{5.5}$$

Step 2: Mean Squared Error of $\hat{A}(D)$ and $\hat{B}(D)$.

As in Step 3 of the proof of theorem 5.2.1, we denote by $\tilde{\Phi}^{-1}_{[\tilde{m},\tilde{M}]} : [\tilde{m},\tilde{M}] \to [-5,5]$ the inverse of Φ over these intervals:

$$\tilde{h}(x) = \begin{cases} \tilde{\Phi}^{-1}_{[\tilde{m},\tilde{M}]}(x) & \text{if } \tilde{m} \leq x \leq \tilde{M} \\ -5 & \text{if } x < \tilde{m} \\ 5 & \text{if } \tilde{M} < x \end{cases}$$

By assumption, $(x,y) \in \mathscr{C}$, $\frac{1}{D}\|A\|_2 \leq 1$ and $\frac{1}{D}\|B\| \leq 1$, $\Delta_{x,y}/\sqrt{2}D \in [-5,5]$ and then, as in Step 3 and 4 of Theorem 5.2.1,

$$\begin{aligned}\mathbb{E}\left(\tilde{h}(F_{x,y}) - \frac{\Delta_{x,y}}{\sqrt{2}D}\right)^2 &\leq \mathbb{E}\left(\tilde{h}(F_{x,y}) - \tilde{h}\left(\Phi\left(\frac{\Delta_{x,y}}{\sqrt{2}D}\right)\right)\right)^2 \\ &\leq L(\tilde{h})^2\mathbb{E}\left(F_{x,y} - \Phi\left(\frac{\Delta_{x,y}}{\sqrt{2}D}\right)\right)^2 \\ &= O(1/n) \text{ by Eq. 5.5.}\end{aligned} \tag{5.6}$$

By applying Eq. 5.6, we then estimate the mean squared error of $\hat{A}(D)$ and $\hat{B}(D)$:

- $\hat{B}_i(D) = \sqrt{2}\tilde{h}(F_{-e_i,e_i})/2$ and $B_i/D = \Delta_{-e_i,e_i}/2D$ $\forall i \in \{1,\dots,d\}$, then $\mathbb{E}(\hat{B}_i(D) - B_i/D)^2 = O(1/n)$ by Eq. 5.6, hence $\mathbb{E}\|\hat{B}(D) - B/D\|^2 = O(1/n)$.

- $\hat{A}_{i,i}(D) = \sqrt{2}\tilde{h}(F_{0,e_i}) - \hat{B}_i(D)$ and $A_{i,i}/D = \Delta_{0,e_i}/D - B_i/D$, then $\mathbb{E}(\hat{A}_{i,i}(D) - A_{i,i}/D)^2 = O(1/n)$ using Eq. 5.6, and

$$\mathbb{E}(\hat{B}_i(D) - B_i/D)^2 = O(1/n).$$

If $i \neq j$, then

$$\hat{A}_{i,j}(D) = \frac{1}{2}\Big(\sqrt{2}\tilde{h}(F_{0,e_i+e_j}) \\ -\hat{B}_i(D) - \hat{A}_{i,i}(D) - \hat{B}_j(D) - \hat{A}_{j,j}(D)\Big),$$

and

$$A_{i,j}/D = \\ 1/2\left(\Delta_{0,e_i+e_j}/D - B_i/D - A_{i,i}/D - B_j/D - A_{j,j}/D\right)$$

hence, by proceeding as above,

$$\mathbb{E}(\hat{A}_{i,j}(D) - A_{i,j}/D)^2 = O(1/n)$$

and

$$\mathbb{E}\|\hat{A}(D) - A/D\|_F^2 = O(1/n).$$

Step 3: with probability at least $1 - O(1/n)$, COPQUAD returns an estimate $\tilde{x}$ solution of $2\tilde{x}\hat{A}(D) = -\hat{B}^t(D)$.

By definition of COPQUAD , $2\tilde{x}\hat{A}(D) \neq -\hat{B}^t(D)$ only if $\tilde{x}$ could not be properly defined because $\hat{A}(D)$ is singular or if we use the projection.

The eigenvalues are continuous (see e.g. [Zedek, 1965]); therefore in a neighbourhood of A/D, $\hat{A}(D)$ has eigenvalues lower bounded by some $\delta > 0$. Therefore, $\hat{A}(D)$ is singular only out of this neighbourhood; this occurs, by Markov's inequality, with probability $O(1/n)$. Therefore, the first case occurs with probability at most $O(1/n)$.

With probability at least $1 - O(1/n)$, the solution $\tilde{x}$ of $2\tilde{x}\hat{A}(D) = -\hat{B}^t(D)$ is therefore the projection of $-\frac{1}{2}\hat{B}(D)^t\hat{A}(D)^{-1}$. For $\hat{A}(D)$ close enough to A/D and $\hat{B}(D)$ close enough to B/D, this is close to x^*, and therefore it is inside $B_d(0, 1-\varepsilon)$.

Step 4: concluding when $2\tilde{x}\hat{A}(D) = -\hat{B}(D)^t$.

Define $B' = B/D - \hat{B}(D)$ and $A' = A/D - \hat{A}(D)$. We have $2x^*A = -B^t$ and $2\tilde{x}\hat{A}(D) = -\hat{B}(D)^t$.

By substraction, we get

$$2(\tilde{x}\hat{A}(D) - x^*A/D) = (B/D)^t - \hat{B}(D)^t$$

i.e. $2(\tilde{x}A/D - \tilde{x}A' - x^*A/D) = B'^t$, using definitions of A' and B'.

By step 2, all terms in A' and B' have expected squared norm $O(1/n)$; and by step 3 $\tilde{x}$ is bounded, therefore

$$2(\tilde{x}A/D - x^*A/D) = B'^t + 2\tilde{x}A'$$

has expected squared norm $O(1/n)$, and

$$(\tilde{x} - x^*) = \frac{1}{2}uA^{-1}D$$

with $\mathbb{E}\|u\|^2 = O(1/n)$.

With $\lambda_{min} > 0$ the smallest eigenvalue of $\frac{1}{D}A$, we get $\mathbb{E}\|\tilde{x} - x^*\|^2 = O(\lambda_{min}^{-2}/n)$.

Note that f can be rewritten as

$$f(x) = (x - x^*)^t A(x - x^*) + C',$$

where $x^* = -\frac{1}{2}B^tA^{-1}$ and $C' = C - x^{*t}Ax^*$.

$$\begin{aligned}\text{Then } SR_n &= \|f(\tilde{x}) - f(x^*)\|^2 = \|(\tilde{x} - x^*)^t A(\tilde{x} - x^*)\|^2 \\ &\leq \lambda_{max}^2 \|\tilde{x} - x^*\|^2\end{aligned}$$

$$\text{Hence } SR_n = O\left(\left(\frac{\lambda_{max}}{\lambda_{min}}\right)^2 \frac{D^2}{n}\right), \text{ which is the expected result.}$$

Step 5: General conclusion

Let us denote by $\mathscr{S}$ the event "COPQUAD returns an estimate $\tilde{x}$ solution of $2\tilde{x}\hat{A}(D) = -\hat{B}(D)^t$" and $\bar{\mathscr{S}}$ its complement. In the following, *diam* denotes the diameter. By definition,

$$\begin{aligned}\mathbb{E}SR_n &= \mathbb{E}(f(\tilde{x}, \omega) - f(x^*, \omega)) \\ &= \underbrace{\mathbb{E}(f(\tilde{x}, \omega) - f(x^*, \omega)|\mathscr{S})}_{=O\left(\left(\frac{\lambda_{max}}{\lambda_{min}}\right)^2 \frac{D^2}{n}\right) \textbf{by step } 4} \underbrace{\mathbb{P}(\mathscr{S})}_{\leq 1} \\ &+ \underbrace{\mathbb{E}(f(\tilde{x}, \omega) - f(x^*, \omega)|\bar{\mathscr{S}})}_{\leq \lambda_{\max}^2 \times D^2 \times diam(B_d(0, 1-\varepsilon))} \underbrace{\mathbb{P}(\bar{\mathscr{S}})}_{=O(1/n) \textbf{by step } 3}\end{aligned}$$

Hence the expected result. □

5.4 Experiments

For each experiment, parameters A, B and C satisfying assumptions in Theorem 5.3.1 are randomly generated. CopQuad then returns an approximation of the optimum of the noisy quadratic function $F(x) = x^t Ax + Bx + C + D\mathscr{G}(0,1)$. Results are obtained over 50 runs.

CopQuad to tackle strong noise. Fig. 5.1 presents results of CopQuad in dimension 2 when the standard deviation D satisfies the assumptions in Theorem 5.3.1, i.e. $\|B\|/D \leq 1$, $|C|/D \leq 1$ and $\|A\|_2/D \leq 1$. The linear rate (in log-log scale) with slope -1 is clearly visible. We obtained similar graphs (not presented here) for dimension 5.

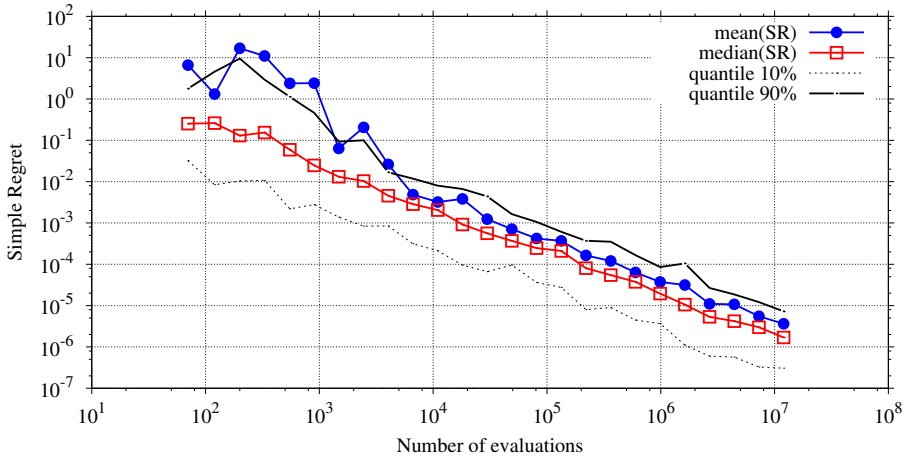
CopQuad with small noise. Figure 5.2 then shows the case of a smaller noise D for dimension 2. Along with the theory ($\|A/D\|_2$ does not satisfy the assumptions), we lose the $O(1/n)$ rate. In the early stages, CopQuad still seems to converge, but it eventually stagnates around the optimum. It is counter-intuitive that an algorithm performs worse when noise decreases; nonetheless, in the case $D \to 0$, the Cop operator always return 0 or 1, thus the estimated parameters are -5 or 5, and the algorithm does not converge. Incidentally, this is consistent with the bandit literature, where the hardest cases are when optimal arms have close values. Providing an algorithm able to cope with $D \leq \|A\|_2$ is possible - asymptotically, as for bandit algorithms mentioned above. Progressively widening the projection interval $[-b(n), b(n)]$ instead of keeping $[-5,5]$ fixed makes this possible; if we have a slow enough function $b : n \mapsto b(n)$ for defining the interval $[-b(n), b(n)]$, then we get:

- e.g. $\log(\log(\log(n)))$ in Eq. 5.6,
- and asymptotically we still get a probability $1/n$ in Step 3 of Theorem 5.3.1.

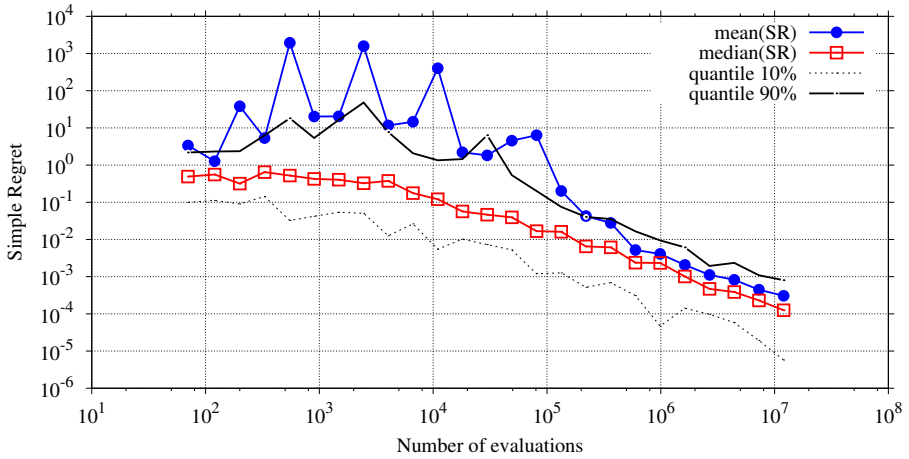
So that, for $n > n_0$, we get Theorem 5.3.1 (up to the slight increase in the bound, depending on the choice of the b function) independently of $D \leq \|A\|_2$ - but n_0 depends on $\frac{1}{D}A$.

5.5 Conclusion

We have shown that comparison-based algorithms can reach a regret $O(1/n)$ on quadratic forms. This partially solves (negatively) a conjecture in [Shamir, 2013], and improves results proposed in [Decock and Teytaud, 2013,



(a) $D = 1$



(b) $D = 10$

Figure 5.1: Dimension $d = 2$, over 50 runs. Mean, median and quantiles 10% and 90% are displayed.

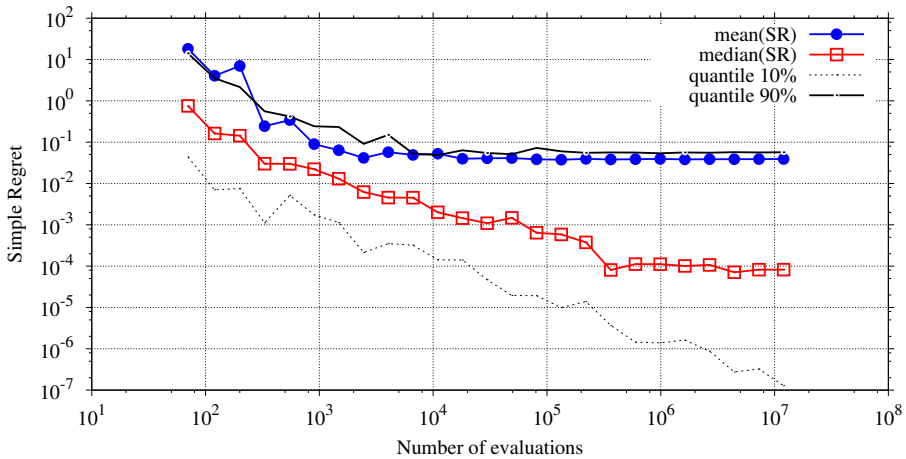


Figure 5.2: $d = 2$, $D = 0.65$.

Rolet and Teytaud, 2010a]. Our main assumption is the Gaussian nature of the noise. We do not assume that the variance is known, but it is supposed to be constant.

Future work. We assume an exactly quadratic function; maybe rates in $O(1/n^{2/3})$ can be reached for non-quadratic functions under smoothness assumptions. Also we might extend the present results to non-Gaussian noise.

Chapter 6

Newton's method: upper bounds

This chapter is based on:

Astete-Morales, S., Cauwet, M.-L., Liu, J., and Teytaud, O. (2015a). Simple and Cumulative Regret for Continuous Noisy Optimization. *Journal of Theoretical Computer Science (TCS)*, 617:12–27

Up to now, we studied comparison-based algorithms. We have shown that Evolution Strategies with a fixed number of resampling converge log-linearly when the noise perturbing the objective function is small enough. We found a lower bound for the Simple Regret on a large family of Evolution Strategies. Last, we proved that it is possible for a comparison-based algorithm to be fast, i.e., to have a Simple Regret in $O(1/n)$. We now turn our attention toward value-based algorithms. Chapter 2 enlightened the thorough results regarding the additive model of noise. We are interested into studying a larger model of noise, defined in Eq. 2.7, that we recall here:

$$f(x,\omega) = f(x) + (f(x) - f(x^*))^z \omega, \tag{2.7}$$

The noise ω satisfies $\mathbb{E}(\omega) = 0$ and

$$Var(f(x,\omega)) = O\left((\mathbb{E}_\omega f(x,\omega) - \mathbb{E}_\omega f(x^*,\omega))^{2z} \right).$$

We focus on the three cases[1] $z = 0$, $z = 0.5$ and $z = 1$. We will refer to them

[1] In the present document, we slightly modified the modelization of [Astete-Morales et al., 2015a] in order to fit the general model of Eq. 2.7. Hence the reader should not be confused by the shift of a factor 2 in all the chapter compared to the original paper.

respectively as the case where the variance of the noise is *constant*, *linear* and *quadratic* as a function of the Simple Regret.

Notations. Depending on the study that one is carrying out, there are arguments for indexing the sequences by *iterations* or by *function evaluations*. In Chapter 2, we adopted the first alternative. It occurs often in the case of optimization of noisy functions, that it is more convenient to have multiple evaluations per iteration. In particular, a classical scheme is to generate a population of search points from a central point at each iteration [Fabian, 1967, Dupač, 1957, Coulom, 2011, Shamir, 2013]. This mechanism is used in the present chapter. Therefore, it is more convenient to introduce the iteration index rather than indexing the sequences by the number of evaluations, when we describe algorithms - but we use indexations by evaluations (i.e. notations described in Chapter 2) when we evaluate convergence rates, and in particular for the slopes of the convergence defined later. We then describe the "dual" notations, with iteration index, and we explain how to switch from an indexation to the other. $x_{m,1},\dots,x_{m,r_m}$ denote the r_m *search points* at iteration m. When we need to access to the mth *evaluated search point*, we define x'_m the mth evaluated search point, i.e. $x'_m = x_{i,k}$ with $m = \sum_{j=1}^{i-1} r_j + k$ and $k \leq r_i$. On the other hand, x_m^{opt}, with only one subscript, is the *recommended point at iteration m*. $\tilde{x}_n$ will always denote the recommendation *after n evaluations*. Hence, when the approximations of the optimum are defined per iteration rather than per evaluation, the sequence of recommended points is redefined as follows: for all $n \geq 1$, $\tilde{x}_n = x_k^{\text{opt}}$, where k is maximal such that $\sum_{i=1}^{k-1} r_i \leq n$.

Criteria. We will consider simultaneously three criteria: the Uniform Rate, the Simple Regret and the Cumulative Regret respectively defined in Eqs. 6.1, 6.2 and 6.3.

$$s(UR) = \limsup_i \frac{\log(UR_i)}{\log(i)} \tag{6.1}$$

$$s(SR) = \limsup_i \frac{\log(SR_i)}{\log(i)} \tag{6.2}$$

$$s(CR) = \limsup_i \frac{\log(CR_i)}{\log(i)} \tag{6.3}$$

where UR_i is the $1-\delta$ quantile of $\|x'_i - x^*\|$, SR_i is the $1-\delta$ quantile of $\mathbb{E}_\omega f(\tilde{x}_i,\omega) - \mathbb{E}_\omega f(x^*,\omega)$, CR_i is the $1-\delta$ quantile of

$\sum_{j\leq i}\Big(\mathbb{E}_\omega f(x'_j,\omega)-\mathbb{E}_\omega f(x^*,\omega)\Big)$. x'_i denotes the ith evaluated search point and $\tilde{x}_i$ denotes the recommendation after i evaluations. We have expectation operators $\mathbb{E}_\omega$ above with respect to ω only, therefore $\mathbb{E}_\omega f(\tilde{x}_i,\omega)$ is not deterministic. Quantiles are with respect to all remaining stochastic parts such as noise in earlier fitness evaluations and possibly internal randomness of the optimization algorithm.

Summary of results. We here extend the state of the art in the case of $z=1$ for all criteria, and $z=0.5$ for more general families of functions (published results were only for sphere functions), and get all the results with the same algorithm. We also generalize existing results for UR or SR or CR to all three criteria. On the other hand, we do not get Fabian's $s(SR)$ arbitrarily close to -1 on smooth non-quadratic functions with enough derivatives, which require a different schema for finite differences and assume the existence of a large number of additional derivatives.

We propose and study a noisy optimization algorithm, which possibly uses a Newton-style approximation, i.e. a local quadratic model. Importantly, our algorithm is not limited to optimization with black-box approximations of gradients and Hessian; we consider more general algorithms with *Low-Squared Error (LSE)* (Def. 6.1.1).

6.1 The Iterative Noisy Optimization Algorithm (INOA)

Section 6.1.1 presents a general iterative algorithm, which uses a *sampling tool* and an *optimum estimator*. It relies on a *Low-Squared Error assumption* (Def. 6.1.1) which is central in the assumptions for the main theorem. Section 6.1.2 provides examples of sampling tools, called SAMPLER functions, and examples of optimum estimators, given by OPT functions, which match the assumptions in Section 6.1.1.

6.1.1 General framework

The Iterative Noisy Optimization Algorithm (INOA) is presented in Alg. 6.1. It uses a pair of functions (SAMPLER, OPT). Specific tasks and properties of these

functions are described below and examples of such functions are given in Section 6.1.2.

Algorithm 6.1 Iterative Noisy Optimization Algorithm (INOA).

Input:

Step-size parameters $\alpha > 0$ and $A > 0$; number of revaluations parameters $\beta \geq 0$ and $B > 0$; initial points $x_1^{\text{opt}} = \tilde{x}_1$; objective function; sampler function $\textsc{Sampler}(\cdot)$; optimizer function $\textsc{Opt}(\cdot)$.

Output:

approximations $(x_n^{\text{opt}})_{n\geq 1}$, recommendations $(\tilde{x}_m)_{m\geq 1}$, evaluation points $(x_{n,i})_{n\geq 1, i\in\{1,\dots,r_n\}}$, fitness evaluations $(y_{n,i})_{n\geq 1, i\in\{1,\dots,r_n\}}$

```
n ← 1
while not finished do
    σ_n = A/n^α                                   ▷ Compute step-size
    r_n = B⌈n^β⌉                                  ▷ Compute revaluations number
    for i = 1 to r_n do
        x_{n,i} = Sampler(x_n^opt, σ_n, i)
        y_{n,i} = fitness evaluation at x_{n,i}
    end for
    x_{n+1}^opt = Opt(x_n^opt, (x_{n,i}, y_{n,i})_{i∈{1,...,r_n}})   ▷ Compute next approximation
    n ← n + 1
end while
```

SAMPLER is the element of the algorithm that provides new search points: given a point x in the search space, SAMPLER provides new search points that lie in the neighborhood of x. More precisely, $\forall i \in \{1,2,\dots\}$, $\textsc{Sampler}(x,\sigma,i)$ outputs a point x_i such that it satisfies $\|x_i - x\| \leq 2\sigma$, with σ a given step-size. Notice that we do not make any assumptions on *how* the new search points are chosen, we only ask for them to be within a given maximal distance from the generator point x. OPT corresponds to the optimum estimator of the algorithm: given $x, x_1, \dots, x_r$ and $y_1, \dots, y_r$ with $y_i = f(x_i, \omega_i)$ (with ω_i independent copies of ω), OPT provides an estimate $x^{\text{opt}} := \textsc{Opt}(x, (x_i, y_i)_{i\in\{1,\dots,r\}})$ of x^*, the arg min of $\mathbb{E}f$. Additionally, for the sake of convergence, the pair $(\textsc{Sampler}, \textsc{Opt})$ verifies a property defined in Def. 6.1.1 and called the *Low-Squared Error assumption*.

The algorithm provides the sequence $(x_n^{\text{opt}})_{n\geq 1}$, indexed with the number of *iterations*, but the recommendations $(\tilde{x}_n)_{n\geq 1}$ in the definitions of Chapter 2 have to be indexed by the number of *evaluations*. Hence, for $m \geq 1$, the recommendation

$\tilde{x}_m$ are defined by $\tilde{x}_m = x_{n(m)}$ with $n(m) = \max\{n; \sum_{i=1}^{n-1} r_i \leq m\}$, since there are r_i evaluations at iteration i.

Definition 6.1.1 (Low-Squared Error (LSE) assumption)**.** *Given a domain $\mathscr{D} \subseteq \mathbb{R}^d$, an objective function $f : \mathscr{D} \to \mathbb{R}$ corrupted by noise. We assume that f is such that $\mathbb{E}_\omega f(x,\omega)$ has a unique optimum x^*. Let $C > 0$, $U > 0$, and $z \in \{0, 0.5, 1\}$. Then, we say that* (SAMPLER, OPT) *has a $(4z-2)$-Low-Squared Error for f, C, U, S if $\forall (r,\sigma) \in S$*

$$\|x - x^*\| \leq C\sigma \implies \textit{for any positive integer } r, \quad \mathbb{E}(\|x^{\text{opt}} - x^*\|^2) \leq U \frac{\sigma^{4z-2}}{r}, \tag{6.4}$$

where x^{opt} is provided by the OPT *function, which receives as input*

- *the given x,*
- *r search points $(x_i)_{i \in \{1,\dots,r\}}$, outputs of* SAMPLER*,*
- *and their corresponding noisy fitness values.*

In the latter definition, z is related to the intensity of the noise. Recall that we consider three types of noise, namely *constant*, *linear* or *quadratic* in function of the *SR*. More precisely, we consider that

$$Var(f(x,\omega)) = O\left([\mathbb{E}_\omega f(x,\omega) - \mathbb{E}_\omega f(x^*,\omega)]^{2z}\right) \text{ with } z \in \{0, 0.5, 1\}.$$

The rate $O(1/r)$ for a squared error is typical in statistics, when estimating some parameters from r samples. We will see in examples below that the scaling with σ is also relevant, as we recover, with the LSE as an intermediate property, many existing rates.

We can work with the additional assumption that $x^* = 0$ without loss of generality. Hence from now on, examples, proofs and theorems are displayed with $x^* = 0$.

6.1.2 Examples of algorithms verifying the LSE assumption

In this section we provide two examples of pairs (SAMPLER, OPT) which verify Def. 6.1.1. Not only SAMPLER and OPT are important, but also the type of functions we consider (conditions for expectation and variance on the properties that

show the verification of LSE). The first example uses an estimation of the gradient of the function to produce an approximation of the optimum. The idea is simple: if we have x, a current approximation to the optimum, we sample around it and use these points to estimate the gradient and obtain the next approximation.

Let $(e_j)_{j=1}^d$ be the standard basis of $\mathbb{R}^d$. SAMPLER will output search points $x \pm \sigma e_j$ for some $j \in \{1, \dots d\}$. Therefore, the set of points that SAMPLER has access to is $E' := E'_+ \cup E'_-$ where $E'_+ = (x + \sigma e_j)_{j=1}^d$, $E'_- = (x - \sigma e_j)_{j=1}^d$ and E' is ordered[2]. In this example, when SAMPLER is queried for the i-th time it will output the i-th point of E'. For the case $i > 2d = |E'|$, to simplify the notation we define a slightly different version of the usual modulo operation, denoted "$\overline{\text{mod}}$", such that for any i, d, $i \ \overline{\text{mod}} \ d = 1 + ((i-1) \bmod d)$. Therefore, when $i > 2d = |E'|$, SAMPLER will output the $(i \ \overline{\text{mod}} \ 2d)$-th point of E'. We assume that SAMPLER outputs at the end a sample of r points, all belonging to E'. Note that as soon as $r > 2d$ the search points are sampled several times. However, the values of the objective function of the same search point evaluated two or more times will differ due to the noise in the evaluation. On the other hand, OPT takes this regular repeated sample around x and its corresponding objective function values to compute an average value for each of the points in E'. Hence, the average is done over at least $\lfloor r/(2d) \rfloor$ function evaluations and it allows to reduce the noise and obtain a more confident - still noisy - evaluation. With these averaged values, OPT computes the approximated optimum. Let us consider

$$Y_{j+} = \{\text{all evaluations of } x + \sigma e_j\} \text{ and } Y_{j-} = \{\text{all evaluations of } x - \sigma e_j\}$$

and use the notation $x^{(j)}$ to refer to the j-th coordinate of x. Also, when we use $\sum Y_{j+}$, with Y_{j+} a set, it will simply denote that we sum over all the elements of the multiset Y_{j+}.

[2] $E' = \{x + \sigma e_1, \dots, x + \sigma e_d, x - \sigma e_1, \dots, x - \sigma e_d\}$

Example 1 Gradient based method verifying the LSE assumption (Def. 6.1.1). Given $x \in \mathbb{R}^d$ and $\sigma > 0$, SAMPLER and OPT are defined as follows.

function SAMPLER(x, σ, i)

$$j \leftarrow i \ \overline{\text{mod}}\ 2d \tag{6.5}$$

$$x_i \leftarrow \text{the } j\text{-th point in } E' \tag{6.6}$$

return x_i
end function

function OPT($x, (x_i, y_i)_{i \in \{1,\dots,r\}}$)
for $j = 1$ to d **do**

$$\hat{y}_{j+} \leftarrow \frac{1}{|Y_{j+}|} \sum Y_{j+}, \qquad \hat{y}_{j-} \leftarrow \frac{1}{|Y_{j-}|} \sum Y_{j-} \tag{6.7}$$

$$\hat{g}^{(j)} \leftarrow \frac{\hat{y}_{j+} - \hat{y}_{j-}}{2\sigma} \tag{6.8}$$

end for

$$x^{\text{opt}} \leftarrow x - \frac{1}{2}\hat{g}$$

return x^{opt}
end function

Property 6.1.1 enunciates the fact that the pair (SAMPLER, OPT) defined in Example 1 satisfies the Low-Squared Error assumption (Def. 6.1.1).

Property 6.1.1. (SAMPLER, OPT) in Example 1 satisfy $(4z-2)$-LSE for the sphere function.

Let f be the function to be optimized, and $z \in \{0, 0.5, 1\}$. We assume that:

Framework 1

$$\mathbb{E}_\omega f(x,\omega) = \|x\|^2 \tag{6.9}$$

$$Var(f(x,\omega)) = O(\|x\|^{4z}) \text{ for some } z \in \{0, 0.5, 1\} \tag{6.10}$$

Then there is $C > 0$, such that if x and σ verify $\|x\| \leq C\sigma$, then

$$\mathbb{E}(\|x^{\text{opt}}\|^2) = O(\sigma^{4z-2}/r). \tag{6.11}$$

where x^{opt} is the output of OPT$(x, (x_i, y_i)_{i \in \{1,\dots,r\}})$, $(x_i)_{i \in \{1,\dots,r\}}$ *is the output of* SAMPLER *and $(y_i)_{i \in \{1,\dots,r\}}$ their respective noisy fitness values.*

Proof. We know that $\mathbb{E}(\|x^{\text{opt}}\|^2) = \sum_{j=1}^{d} \left\{ Var(x^{\text{opt}(j)}) - (\mathbb{E}(x^{\text{opt}(j)}))^2 \right\}$. For all $j \in \{1, \dots d\}$, using the definition of $\hat{g}^{(j)}$ in Eq. 6.8 and using Eq. 6.9 we obtain

$$\mathbb{E}(\hat{g}^{(j)}) = 2x^{(j)} \Rightarrow \mathbb{E}(x^{\text{opt}(j)}) = 0.$$

Now, using the variance of the noisy function in Eq. 6.10 and the fact that $\|x\| \leq C\sigma$,

$$Var(\hat{g}^{(j)}) = O\left(\frac{\sigma^{4z-2}}{r}\right) \Rightarrow \mathbb{E}(\|x^{\text{opt}}\|^2) = O\left(\frac{\sigma^{4z-2}}{r}\right).$$

□

The method using gradients described above is already well studied, as well as improved variants of it with variable step-sizes, (see [Fabian, 1967, Chen, 1988, Shamir, 2013]).

Therefore, we now switch to the second example, including the computation of the Hessian. As in the Example 1, we consider a set of search points that are available for SAMPLER to output. Let us define $E'' = \{x \pm \sigma e_i \pm \sigma e_j; 1 \leq i < j \leq d\}$. And so the sample set will be E, which includes the set E'' defined above and the sample set E' defined for Example 1. Therefore, $|E| = 2d^2$ (E' has cardinal $2d$ and E'' has cardinal $2d(d-1)$). Also, we define naturally the sets of evaluations of the search points as follows:

$$\begin{aligned}
Y_{j+,k+} &= \{\text{all evaluations of } x + \sigma e_j + \sigma e_k\}, \\
Y_{j+,k-} &= \{\text{all evaluations of } x + \sigma e_j - \sigma e_k\}, \\
Y_{j-,k+} &= \{\text{all evaluations of } x - \sigma e_j + \sigma e_k\}, \\
Y_{j-,k-} &= \{\text{all evaluations of } x - \sigma e_j - \sigma e_k\}.
\end{aligned}$$

Example 2 Noisy-Newton method verifying the $(4z-2)$-LSE assumption. Given $x \in \mathbb{R}^d$, $\sigma > 0$ and $c_0 > 0$, SAMPLER and OPT are defined as follows. M^t denotes the transpose of matrix M.

function SAMPLER(x, σ, i)

$$j \leftarrow i \overline{\mod} 2d^2 \tag{6.12}$$

$$x_i \leftarrow \text{the } j\text{-th point in } E \tag{6.13}$$

return x_i
end function

---

function OPT($x, (x_i, y_i)_{i \in \{1,\ldots,r\}}$)
for $j = 1$ to d **do**

$$\hat{y}_{j+} \leftarrow \frac{1}{|Y_{j+}|} \sum Y_{j+}, \quad \hat{y}_{j-} \leftarrow \frac{1}{|Y_{j-}|} \sum Y_{j-} \tag{6.14}$$

$$\hat{g}^{(j)} \leftarrow \frac{\hat{y}_{j+} - \hat{y}_{j-}}{2\sigma} \tag{6.15}$$

end for
for $1 \le j,k \le d$ **do**

$$\hat{y}_{j+,k+} \leftarrow \frac{1}{|Y_{j+,k+}|} \sum Y_{j+,k+}, \qquad \hat{y}_{j+,k-} \leftarrow \frac{1}{|Y_{j+,k-}|} \sum Y_{j+,k-}$$

$$\hat{y}_{j-,k+} \leftarrow \frac{1}{|Y_{j-,k+}|} \sum Y_{j-,k+}, \qquad \hat{y}_{j-,k-} \leftarrow \frac{1}{|Y_{j-,k-}|} \sum Y_{j-,k-}$$

$$\hat{h}^{(j,k)} \leftarrow \frac{\left(\hat{y}_{j+,k+} - \hat{y}_{j-,k+}\right) - \left(\hat{y}_{j+,k-} - \hat{y}_{j-,k-}\right)}{4\sigma^2}$$

end for

$$\hat{h} \leftarrow \frac{\hat{h} + \hat{h}^t}{2}$$

if $\hat{h}$ is positive definite with least eigenvalue greater than c_0 **then**

$$x^{\text{opt}} \leftarrow x - (\hat{h})^{-1} \hat{g} \tag{6.16}$$

else

$$x^{\text{opt}} \leftarrow x \tag{6.17}$$

end if
return x^{opt}
end function

Note that in Example 2, the output of $\textsc{Sampler}(x, \sigma, i)$ are equally distributed over E so that each of them is evaluated at least $\lfloor r/2d^2 \rfloor$ times. The pair (SAMPLER, OPT) defined in Example 2 verifies the LSE assumption (Property 6.1.2) when the noisy objective function is approximately quadratic (Eq 6.18) and the noise follows the constraint given by Eq. 6.19.

Property 6.1.2. **(SAMPLER, OPT) in Example 2 satisfy LSE.** *Let f be the function to be optimized, $z \in \{0, 0.5, 1\}$. We assume that:*

Framework 2

$$\mathbb{E}_\omega f(x,\omega) = \sum_{1\le j,k\le d} c_{j,k} x^{(j)} x^{(k)} + \sum_{1\le j,k,l\le d} b_{j,k,l} x^{(j)} x^{(k)} x^{(l)} + o(\|x\|^3), \text{ with } c_{j,k} = c_{k,j} \tag{6.18}$$

$$Var(f(x,\omega)) = O(\|x\|^{4z}) \text{ where } z \in \{0, 0.5, 1\}. \tag{6.19}$$

Assume that there is some $c_0 > 0$ such that h is positive definite with least eigenvalue greater than $2c_0$, where h is the Hessian of $\mathbb{E}f$ at 0, i.e. $h = (2c_{j,k})_{1\le j,k\le d}$.

Then there exists $\sigma_0 > 0$, $K > 0$, $C > 0$, such that for all σ that satisfies

$$(i)\ \ \sigma < \sigma_0 \ \text{ and } (ii)\ \ \sigma^{6-4z} \le K/r,$$

and for all x such that

$$\|x\| \le C\sigma, \tag{6.20}$$

we have

$$\mathbb{E}\,\|x^{\text{opt}}\|^2 = O\left(\frac{\sigma^{4z-2}}{r}\right),$$

where x^{opt} is the output of $\textsc{Opt}(x, (x_i, y_i)_{i=1}^r)$, $(x_i)_{i=1}^r$ are the output of $\textsc{Sampler}(x, \sigma, i)$ and the $(y_i)_{i=1}^r$ are their respective noisy fitness values.

Proof. The event $\mathscr{E}_{\hat{h}}^{c_0}$ denotes the fact that the matrix $\hat{h}$ is positive definite with least eigenvalue greater than c_0, and $\overline{\mathscr{E}_{\hat{h}}^{c_0}}$ is the complementary event.

$$\mathbb{E}\,\|x^{\text{opt}}\|^2 = \underbrace{\mathbb{E}(\|x^{\text{opt}}\|^2|\overline{\mathscr{E}_{\hat{h}}^{c_0}})}_{A_1}\underbrace{\mathbb{P}(\overline{\mathscr{E}_{\hat{h}}^{c_0}})}_{A_2} + \underbrace{\mathbb{E}(\|x^{\text{opt}}\|^2|\mathscr{E}_{\hat{h}}^{c_0})}_{A_3}\underbrace{\mathbb{P}(\mathscr{E}_{\hat{h}}^{c_0})}_{\le 1}$$

where $A_1 \le (C\sigma)^2$ using Eqs. 6.17 and 6.20; $A_2 = O\left(\frac{\sigma^{4z-2}}{r\sigma^2}\right)$ by Lemma C.0.2; $A_3 = O\left(\frac{\sigma^{4z-2}}{r}\right)$ by Lemma C.0.3. Therefore $\mathbb{E}\,\|x^{\text{opt}}\|^2 = O\left(\frac{\sigma^{4z-2}}{r}\right)$, which is the expected result. □

Remark 6.1.1. *Using the expressions of σ and r given by INOA, if $(6-4z)\alpha \geq \beta$, and given $A > 0$, then there exists a constant $B_0 > 0$ such that if $B > B_0$ then the condition $\sigma_n^{6-4z} \leq K/r_n$ is satisfied.*

6.2 Convergence rates of INOA

Sections 6.2.1 and 6.2.2 provide, respectively, the main result and its applications, namely Cumulative Regret analysis and Simple Regret analysis for various models of noise. The special case of twice-differentiable functions is studied in Section 6.2.3.

6.2.1 Rates for various noise models

In this section, we present the main result, i.e. the convergence rates of INOA.

Theorem 6.2.1 (Rates for various noise models)**.** *Consider some $A > 0$ and consider the iterative noisy optimization algorithm (INOA, Alg. 6.1, with parameters A, B, α, β). Assume that $(\text{SAMPLER}, \text{OPT})$ has a $(4z-2)$-Low-Squared Error assumption (LSE, Def. 6.1.1) for some f, C, U, S. Assume that $B > B_0$, where B_0 depends on α, β and A only. Let us assume that INOA provides (r_n, σ_n) always in S, and let us assume that*

$$1 < \beta + \alpha(4z-4). \tag{6.21}$$

Consider $\delta > 0$. Then there is $C > 0$, such that if $x_1^{\text{opt}} = \tilde{x}_1$ satisfies $\|x_1^{\text{opt}}\| \leq CA$, then with probability at least $1-\delta$,

$$\forall n, \quad \|x_n^{\text{opt}}\| \leq C\sigma_n, \tag{6.22}$$

$$\forall n, \forall i \leq r_n, \quad \|x_{n,i}\| \leq (C+2)\sigma_n. \tag{6.23}$$

Remark 6.2.1. *It is assumed that given x,* SAMPLER *provides a new search point x_i such that $\|x_i - x\| \leq 2\sigma$ (see Section 6.1.1). This together with Eq. 6.22 gives $\|x_{n,i}\| \leq \|x_{n,i} - x_n^{\text{opt}}\| + \|x_n^{\text{opt}}\| \leq (C+2)\sigma_n$. Hence Eq. 6.23 holds if Eq. 6.22 holds; we just have to show Eq. 6.22.*

General organization of the proof of Eq. 6.22: Assume that Eq. 6.21 holds. Consider a fixed $C > 0$ and $1 > \delta > 0$. Consider hypothesis H_n: for any $1 \leq i \leq n$, $\|x_i^{\text{opt}}\| \leq C\sigma_i$ with probability at least $1-\delta_n$, where

$$\delta_n = \sum_{i=1}^{n} c i^{-\beta-\alpha(4z-4)}. \tag{6.24}$$

c is chosen such that $\forall n \geq 1$, $\delta_n \leq \delta$. By Eq. 6.21, $\sum_{i=1}^{\infty} i^{-\beta-\alpha(4z-4)} = \Delta < \infty$, and $c = \delta/\Delta$ is suitable. We prove that for any positive integer n, H_n holds. The proof is by induction on H_n. H_1 is true since x_1^{opt} is chosen such that $\|x_1^{\text{opt}}\| \leq CA$, i.e. $\|x_1^{\text{opt}}\| \leq C\sigma_1$.

Proof. Assume that H_n holds for a given integer n. We will show that H_{n+1} holds.

Step 1: concentration inequality for x_{n+1}.

By design of INOA, Alg. 6.1, Line 9, $x_{n+1}^{\text{opt}} = \textsc{Opt}(x_n^{\text{opt}}, (x_{n,i}, y_{n,i})_{i=1}^{r_n})$. When H_n is true, with probability at least $1-\delta_n$, $\|x_n^{\text{opt}}\| \leq C\sigma_n$.

This together with the LSE imply that conditionally to an event with probability at least $1-\delta_n$,

$$\begin{aligned}
\mathbb{E}(\|x_{n+1}^{\text{opt}}\|^2) &\leq U\frac{\sigma_n^{4z-2}}{r_n} \\
&\leq U\left(\frac{A}{n^\alpha}\right)^{4z-2}\frac{1}{B\lceil n^\beta\rceil} \\
&\leq U\frac{A^{4z-2}}{B}\left(\frac{n}{n+1}\right)^{-\alpha(4z-2)-\beta}(n+1)^{-\alpha(4z-2)-\beta} \\
&\leq M(n+1)^{-\alpha(4z-2)-\beta} \qquad (6.25) \\
\text{where } M &= U\frac{A^{4z-2}}{B}\left(\sup_{n\geq 1}(\frac{n}{n+1})^{-\alpha(4z-2)-\beta}\right). \qquad (6.26)
\end{aligned}$$

Step 2: applying Markov's inequality. By Markov's inequality,

$$\mathbb{P}\left(\|x_{n+1}^{\text{opt}}\| > C\sigma_{n+1}\right) = \mathbb{P}\left(\|x_{n+1}^{\text{opt}}\|^2 > C^2\sigma_{n+1}^2\right) \leq \frac{\mathbb{E}\|x_{n+1}^{\text{opt}}\|^2}{C^2\sigma_{n+1}^2}.$$

We apply Eq. 6.25:

$$\begin{aligned}
\mathbb{P}\left(\|x_{n+1}^{\text{opt}}\| > C\sigma_{n+1}\right) &\leq \frac{M}{C^2A^2}(n+1)^{\alpha(2-(4z-2))-\beta} \\
&\leq c(n+1)^{-\beta-\alpha(4z-4)} = \varepsilon_{n+1} \text{ if } B > B_0,
\end{aligned}$$

where $B_0 = \frac{UA^{4z-4}\left(\sup_{n\geq 1}(\frac{n}{n+1})^{-\alpha(4z-2)-\beta}\right)}{cC^2}$, using Eq. 6.26. Then, with probability $(1-\delta_n)(1-\varepsilon_{n+1})$, $\|x_{n+1}^{\text{opt}}\| \leq C\sigma_{n+1}$. Hence with probability at least $1-\delta_n-\varepsilon_{n+1} = 1-\delta_{n+1}$, $\|x_{n+1}^{\text{opt}}\| \leq C\sigma_{n+1}$. This is H_{n+1}. The induction is complete. □

6.2.2 Application: the general case

Theorem 6.2.1 ensures some explicit convergence rates for *SR* and *CR* depending on parameters α, β and z.

Corollary 6.2.1. *Consider the context and assumptions of Theorem 6.2.1, including some* (SAMPLER, OPT) *which has a* $(4z-2)$*-LSE (Def. 6.1.1) for some* f, C, U, S *such that for all* n, $(r_n, \sigma_n) \in S$, *and let us assume that* $\mathbb{E}_w f(x,\omega) - \mathbb{E}_\omega f(x^*,\omega) = O(\|x-x^*\|^2)$.

Then, the Simple Regret of INOA of has slope $s(SR) \leq \frac{-\alpha(4z-2)-\beta}{\beta+1}$ *and the Cumulative Regret has slope* $s(CR) \leq \frac{\max(0,1+\beta-2\alpha)}{1+\beta}$.

Quadratic case: in the special case $z=0$ *and if* $\mathbb{E}f$ *is quadratic (i.e.* $\mathbb{E}_\omega f(x,\omega) = \sum_{1\leq j,k\leq d} c_{j,k} x^{(j)} x^{(k)}$*), we get* $s(SR) \leq \frac{2\alpha-\beta}{\beta+1}$.

Proof. The number of evaluations until the end of iteration n, before recommending x^{opt}_{n+1}, is $m(n) = \sum_{i=1}^n r_i = O(n^{\beta+1})$.

- By assumption, $\mathbb{E}_\omega f(x,\omega) - \mathbb{E}_\omega f(x^*,\omega) = O(\|x-x^*\|^2)$. Markov's inequality applied to $\|x-x^*\|^2$ gives: $\mathbb{P}\left(\|x-x^*\|^2 > \frac{\mathbb{E}\|x-x^*\|^2}{\delta}\right) < \delta$. Hence, the Simple Regret SR_n after iteration n, when recommending $\tilde{x}_{m(n)} = x^{\text{opt}}_{n+1}$, is the $1-\delta$ quantile of $\|x^{\text{opt}}_{n+1} - x^*\|^2$, this is $O\left(\mathbb{E}\|x^{\text{opt}}_{n+1} - x^*\|^2\right)$. Using step 1 of Theorem 6.2.1, it follows that $SR_n = O\left(n^{-(4z-2)\alpha-\beta}\right)$.

- the Cumulative Regret CR_n until iteration n is the $1-\delta$ quantile of $\sum_{1\leq i\leq r_m, 1\leq m\leq n} \mathbb{E}_\omega f(x_{i,m},\omega) - \mathbb{E}_\omega f(x^*,\omega) = \sum_{1\leq i\leq r_m, 1\leq m\leq n} O\left(\|x_{i,m} - x^*\|^2\right)$. By Theorem 6.2.1, Eq. 6.23:

$$
\begin{aligned}
O\left(\sum_{i=1}^n r_i (C+2)^2/i^{2\alpha}\right) &= O\left(\sum_{i=1}^n i^\beta (C+2)^2/i^{2\alpha}\right) \\
&= \begin{cases} O\left(n^{1+\beta-2\alpha}\right) & \text{if } \beta-2\alpha > -1, \\ O(\log(n)) & \text{if } \beta-2\alpha = -1 \\ O(1) & \text{otherwise.} \end{cases}
\end{aligned}
$$

Dividing the log of Simple Regret at iteration n by the logarithm of the number of evaluations until iteration n leads to the expected result (slope) for Simple Regret.

Dividing the log of Cumulative Regret until iteration n by the logarithm of the number of evaluations until iteration n leads to the expected result for Cumulative Regret.

□

6.2.3 Application: the smooth case

Table 6.1 presents optimal $s(SR)$ and $s(CR)$ in the more familiar case of smooth functions, with at least two derivatives. All results in this table can be obtained by INOA with OPT and SAMPLER as in Example 2 and the provided parametrizations for α and β, except the result by [Fabian, 1967] assuming many derivatives.

In all cases except the quadratic case with $z = 0$, we assume $(6-4z)\alpha > \beta$, so that the LSE assumption holds for INOA with OPT and SAMPLER as in Example 2 (see Property 6.1.2) and we assume $1 < \beta + \alpha(4z-4)$ so that Eq. 6.21 in Theorem 6.2.1 holds. Regarding the special case of $z = 0$ and quadratic function, the equation to satisfy is $1 < \beta - 4\alpha$. Please note that in this last case, the assumption $(6-4z)\alpha > \beta$ is not necessary. We then find out values of α and β such that good slopes can be obtained for CR and SR. Algorithms ensuring a slope $s(CR)$ in this table also ensure a slope $s(UR) = \frac{1}{2}(s(CR)-1)$. It follows that the optimal parametrization for UR is the same as the optimal parametrization for CR.

We consider parameters optimizing the CR (left) or SR (right) - and both simultaneously when possible. These results are for B constant but large enough. Infinite values mean that the value can be made arbitrarily negatively large by choosing a suitable parametrization. X^+ denotes a value which should be made arbitrarily close to X by superior values, in order to approximate the claimed rate.

Results are not adaptive; we need a different parametrization when $z = 0$, $z = 0.5$, $z = 1$. Also, for $z = 0$, we need a different parametrization depending on whether we are interested in CR or SR.

6.3 Conclusion

We have shown that estimating the Hessian and gradient can lead to fast convergence results. In fact, with one unique algorithm we obtain many of the rates presented by

- [Spall, 2000, Shamir, 2013] in the case of a constant variance noise for Simple Regret and Cumulative Regret respectively.

Table 6.1: $s(SR)$ and $s(CR)$ for INOA for various values of α and β, in the case of twice-differentiable functions. The references mean that our algorithm gets the same rate as in the cited paper. No reference means that the result is new.

z	optimized for CR		optimized for SR	
	$s(SR)$	$s(CR)$	$s(SR)$	$s(CR)$
0 (constant var)	$\alpha \simeq \infty, \beta \simeq 4\alpha + 1^+$		$\beta = 6\alpha, \alpha = \infty$	
	$-1/2$	$1/2$ [Fabian, 1967] [Shamir, 2013]	$-2/3$ [Dupač, 1957]	$2/3$
0 and ∞-differentiable			-1 [Fabian, 1967] [Polyak and Tsybakov, 1990]	
0 and quadratic			$\alpha = 0, \beta \simeq \infty$	
			-1 [Dupač, 1957]	
0.5 (linear var)	$\alpha \simeq \infty, \beta \simeq 2\alpha + 1^+$			
	-1 [Rolet and Teytaud, 2010b]	0	-1	0
1 (quadratic var)	$\alpha \simeq \infty, \beta > 1$			
	$-\infty$	0	$-\infty$	0

- [Coulom et al., 2011] ($z = 0.5$) and [Auger, 2005] ($z = 1$) for a larger space of functions than in these papers, where sphere functions are considered.

In summary, we observe in Table 6.1 that results obtained here recover most previous results discussed in Chapter 2. And also the results presented here cover all the analysed criteria: Simple Regret, Cumulative Regret, Uniform Rate.

Compared to [Spall, 2000, Polyak and Tsybakov, 1990], our algorithm uses more evaluations per iteration. This has advantages and drawbacks. The positive part is that it is therefore more parallel. For example, for $z = 0$, and an algorithm optimized for SR, we get $s(SR) = -2/3$; this rate is the same as the one in [Spall, 2000] in terms of number of evaluations, i.e. the number of evaluations is proportional to $(1/sr)^{2/3}$ for a Simple Regret sr, but our evaluations are grouped into a small number of iterations. On the other hand, it is far less convenient in a sequential setting as the optimization process starts only after an iteration is complete, which takes a significant time in our case. Our algorithm is proved for $z = 0.5$, $z = 1$; these cases are not discussed in [Shamir, 2013, Fabian, 1967, Spall, 2000, Polyak and Tsybakov, 1990].

Our algorithm is not limited to functions with quadratic approximations; quadratic approximations are a natural framework, but the success of various surrogate models in recent years suggests that other approximation frameworks could be used. Our theorems are not specific for quadratic approximations and only require that the LSE approximation holds. The LSE assumption is natural in terms of scaling with respect to r - the $1/\sqrt{r}$ typical deviation is usual in e.g. maximum likelihood estimates, and therefore the method should be widely applicable for general surrogate models.

More generally, our results show a fast rate as soon as the estimator of the location of the optimum has squared error $O(\sigma^{4z-2}/r)$, when using r points sampled adequately within distance $O(\sigma)$ of the optimum.

Further work. On the theoretical side, further work includes writing detailed constants, in particular depending on the eigenvalues of the Hessian of the expected objective function at the optimum and the dimension of the search space. In the case of infinite slope (see Table 6.1, $z = 1$), we conjecture that the convergence is log-linear, i.e. the logarithm of the Simple Regret decreases as a function of the number of evaluations. On the other hand, future study consists of extensive experiments - but we refer to [Cauwet et al., 2014, Cauwet et al., 2016b] for significant artificial experiments and [Liu and Teytaud, 2014] for the application which motivated this work.

Part of the agenda is to extend the algorithm by providing other examples of estimators to be used for approximating the location of the optimum (other than Examples 1 and 2, but verifying the LSE assumption); in particular, classical surrogate models, and applications to piecewise linear strongly convex functions as in [Rolet and Teytaud, 2010a]. A way to improve the algorithm is to use quasi-Newton estimates of the Hessian, from the successive gradients, rather than using directly finite differences. Last, making algorithms more adaptive by replacing the constants by adaptive parameters depending on noise estimates is under consideration.

Chapter 7

Algorithm Portfolio

This chapter is based on:

Cauwet, M.-L., Liu, J., and Teytaud, O. (2014). Algorithm portfolios for noisy optimization: Compare solvers early. In Pardalos, M. P., Resende, G. M., Vogiatzis, C., and Walteros, L. J., editors, *Learning and Intelligent Optimization: 8th International Conference, Lion 8, Gainesville, FL, USA, February 16-21, 2014. Revised Selected Papers*, pages 1–15. Springer International Publishing, Cham

Cauwet, M.-L., Liu, J., Rozière, B., and Teytaud, O. (2016b). Algorithm portfolios for noisy optimization. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):143–172

So far, we have studied a wide range of algorithms, showing that both families - comparison-based and value-based algorithms - can reach optimal performance. Their efficiency depends on some parameters, which change depending on the parameters of the objective function and on the model of noise. Optimization algorithms might also be sensitive to the starting point. Given a noisy optimization black-box problem, it is then natural to wonder which algorithm to use. In the absence of information about the characteristic of the objective function, should we choose a variant of an Evolution Strategy or Fabian's algorithm, Polyak-Tsybakov's algorithm? And which parameters should we choose? We address these questions through the use of a *portfolio*.

A portfolio of solvers is a set of solvers equipped with an algorithm selection tool for distributing the computational power among them. Portfolios are widely and successfully used in combinatorial optimization [Kotthoff, 2014].

In this work, we study portfolios of noisy optimization solvers. We obtain mathematically proved performance (in the sense that the portfolio performs nearly as well as the best of its solvers) by an ad hoc portfolio algorithm dedicated to noisy optimization. A somehow surprising result is that it is better to compare solvers with some *lag*, i.e., propose the current recommendation of best solver based on their performance *earlier* in the run. An additional finding is a principled method for distributing the computational power among solvers in the portfolio.

Algorithm Selection (AS) consists in choosing, in a portfolio of solvers, the one which is approximately the most efficient on the problem at hand. AS can mitigate the difficulties for choosing a priori the best solver among a portfolio of solvers. This means that AS leads to an adaptive version of the algorithms. In some cases, AS outperforms all individual solvers by combining the good properties of each of them, with information sharing or with chaining, as discussed later. It can also be used for the sake of parallelization or parameter tuning, or for mitigating the impact of bad luck in randomized solvers. In this chapter, we apply AS to the black-box noisy optimization problem.

7.1 Algorithm selection

Combinatorial optimization is probably the most classical application domain for AS [Kotthoff, 2014]. However, machine learning is also a classical test case [Utgoff, 1989]; in this case, AS is sometimes referred to as meta-learning [Aha, 1992].

No free lunch. [Wolpert and Macready, 1997] proves that it is not possible to do better, on average (uniform average) on all optimization problems from a given finite domain to a given finite codomain. This implies that no AS can outperform existing algorithms on average on this uniform probability distribution of problems. Nonetheless, reality is very different from a uniform average of optimization problems, and AS does improve performance in many cases.

Chaining and information sharing. Algorithm chaining [Borrett and Tsang, 1996] means switching from one solver to another during the AS run. More generally, a *hybrid* algorithm is a combination of existing algorithms by any means [Vassilevska et al., 2006]. This is an extreme case of

sharing. Sharing consists in sending information from some solvers to other solvers; they communicate in order to improve the overall performance.

Static portfolios & parameter tuning. A portfolio of solvers is usually static, i.e., combines a finite number of given solvers. SATzilla is probably the most well known portfolio method, combining several SAT-solvers [Xu et al., 2008]. Samulowitz and Memisevic have pointed out in [Samulowitz and Memisevic, 2007] the importance of having "orthogonal" solvers in the portfolio, so that the set of solvers is not too large, but approximates as far as possible the set of all feasible solvers. AS and parameter tuning are combined in [Xu et al., 2011]; parameter tuning can be viewed as an AS over a large but structured space of solvers. We refer to [Kotthoff, 2014] and references therein for more information on parameter tuning and its relation to AS; this is beyond the scope of this chapter.

Fair or unfair distribution of computation budgets. In [Pulina and Tacchella, 2009], different strategies are compared for distributing the computation time over different solvers. The first approach consists in running all solvers for a finite time, then selecting the best performing one, and then keeping it for all the remaining time. Another approach consists in running all solvers with the same time budget independently of their performance on the problem at hand. Surprisingly enough, they conclude that uniformly distributing the budget is a good and robust strategy. The situation changes when a training set is available, and when we assume that the training set is relevant for the future problems to be optimized; [Kadioglu et al., 2011], using a training set of problems for comparing solvers, proposes to use 90% of the time allocated to the best performing solver, the other 10% being equally distributed among other solvers. In [Gagliolo and Schmidhuber, 2005, Gagliolo and Schmidhuber, 2006], it is proposed to use 50% of the time budget for the best solver, 25% for the second best, and so on. Some AS algorithms [Gagliolo and Schmidhuber, 2006, Armstrong et al., 2006] do not need a separate training phase, and perform entirely online solver selection; a weakness of this approach is that it is only possible when a large enough budget is available, so that the training phase has a minor cost. A portfolio algorithm, namely Noisy Optimization Portfolio Algorithm (NOPA), designed for noisy optimization solvers, and which distributes uniformly the computational power among them, is proposed in [Cauwet et al., 2014]. We extend it to Improved Noisy Optimization Portfolio Algorithm (INOPA), which is allowed to distribute non-uniformly the

budget. It is proved that INOPA reaches the same convergence rate as the best solver, within a small (converging to 1) multiplicative factor on the number of evaluations, when there is a unique optimal solver - thanks to a principled distribution of the budget into (i) running all the solvers; (ii) comparing their results; (iii) running the best performing one. The approach is anytime, in the sense that the computational budget does not have to be known in advance.

Parallelism. We refer to [Hamadi, 2013] for more on parallel portfolio algorithms (though not in the noisy optimization case). Portfolios can naturally benefit from parallelism; however, the situation is different in the noisy case, which is highly parallel by nature (as noise is reduced by averaging multiple resamplings, see Chap. 2 Section 2.1.3).

Best solver first. [Pulina and Tacchella, 2009] point out the need for a good ordering of solvers, even if it has been decided to distribute nearly uniformly the time budget among them: this improves the anytime behavior. For example, they propose, within a given scheduling with same time budget for each optimizer, to use first the best performing solver. We will adapt this idea to our context; this leads to INOPA, improved version of NOPA.

Bandit literature. During the last decade, a wide literature on bandits [Lai and Robbins, 1985, Auer, 2002, Bubeck et al., 2009] has proposed many tools for distributing the computational power over stochastic options to be tested. The application to our context is however far from being straightforward. In spite of some adaptations to other contexts (time varying as in [Kocsis and Szepesvári, 2006] or adversarial [Grigoriadis and Khachiyan, 1995, Auer et al., 1995]), and maybe due to strong differences such as the very non-stationary nature of bandit problems involved in optimization portfolios, these methods did not, for the moment, really find their way to AS. Another approach consists in writing this bandit algorithm as a meta-optimization problem; [St-Pierre and Liu, 2014] applies the differential evolution algorithm [Storn and Price, 1995] to some non-stationary bandit problem, which outperforms the classical bandit algorithm on an AS task.

The main contributions of this chapter can be summarized as follows. First, we prove positive results for a portfolio algorithm, termed NOPA, for AS in noisy optimization. Second, we design a new AS, namely INOPA, which (i) gives the priority to the best solvers when distributing the computational power; (ii) ap-

proximately reaches the same performance as the best solver; (iii) possibly shares information between the different solvers. We then prove the requirement of selecting the solver that was apparently the best *some time before* the current iteration - a phenomenon that we term the *lag*. Finally, we provide some experimental results.

7.2 Algorithms

Section 7.2.1 introduces some notations. Section 7.2.2 provides some background and criteria. Section 7.2.3 describes two portfolio algorithms, one with fair distribution of budget among solvers and one with unfair distribution of budget. Section 7.3 then provides theoretical guarantees.

7.2.1 Notations

We recall that (see Appendix A) $\mathbb{N}^* = \{1,2,3,\dots\}$. $o(.), O(.), \Theta(.)$ are the standard Landau notations. $\mathscr{G}$ denotes a standard Gaussian distribution, in dimension given by the context.

The noisy objective function is denoted by $f(\cdot,\omega)$ and its optimum x^*. $\hat{\mathbb{E}}_s[f(x,\omega)]$ denotes the empirical evaluation of $\mathbb{E}[f(x,\omega)]$ over $s \in \mathbb{N}^*$ resamplings, i.e., $\hat{\mathbb{E}}_s[f(x,\omega)] = \frac{1}{s}\sum_{j=1}^{s}\big(f(x,\omega_j)\big)$, where $(\omega_1,\dots,\omega_s)$ denotes a sample of independent identically distributed random variables, copies of ω.

7.2.2 Definitions and criteria

Simple Regret criterion for portfolio. For a portfolio algorithm in the black-box setting, $\forall i \in \{1,\dots,M\}$, $\tilde{x}_{i,n}$ denotes the point

- that the solver number i *recommends* as an approximation of the optimum;
- after this solver has spent n evaluations from the budget.

The Simple Regret corresponding to solver number i after n evaluations (i.e., after solver number i has spent n evaluations), is denoted by $SR_{i,n} := \mathbb{E}(f(\tilde{x}_{i,n},\omega) - f(x^*,\omega))$. For $n \in \mathbb{N}^*$, i_n^* denotes the solver chosen by the selection algorithm when there are at most n evaluations per solver.[1]

[1] This is not uniquely defined, as there might be several time steps at which the maximum number of evaluations in a solver is n; however, the results in the rest of this chapter are independent of this subtlety.

Another important concept is the difference between the two kinds of terms in the regret of the portfolio. We distinguish these two kinds of terms in the next two definitions:

Definition 7.2.1 (Solvers' regret)**.** *The solvers' regret with index n, denoted by $SR_n^{Solvers}$, is the minimum Simple Regret among the solvers after at most n evaluations each, i.e., $SR_n^{Solvers} = \min_{i \in \{1,\dots M\}} SR_{i,n}$.*

Definition 7.2.2 (Selection regret)**.** *The selection regret with index n, denoted by $SR_n^{Selection}$ includes the additional regret due to mistakes in choosing among these M solvers after at most n evaluations each, i.e., $SR_n^{Selection} = \mathbb{E}\left(f(\tilde{x}_{i_n^*,n}, \omega) - f(x^*, \omega)\right)$.*

Similarly, $\Delta_{i,n}$ quantifies the regret for choosing solver i at iteration n.

Definition 7.2.3. *For any solver $i \in \{1,\dots,M\}$ and any number of evaluations $n \in \mathbb{N}^*$, we denote by $\Delta_{i,n}$ the quantity: $\Delta_{i,n} = SR_{i,n} - \min_{j \in \{1,\dots,M\}} SR_{j,n}$.*

Finally, we consider a function that will be helpful for defining our portfolio algorithms.

Definition 7.2.4 (LAG function)**.** *A lag function $\text{LAG} : \mathbb{N}^* \to \mathbb{N}^*$ is a non-decreasing function such that for all $n \in \mathbb{N}^*$, $\text{LAG}(n) \leq n$.*

7.2.3 Portfolio algorithms

In this section, we present two AS methods. A first version shares the computational budget uniformly; a second version has an unfair sharing of computational budget.

Simple Case: Uniform Portfolio NOPA

We present in Algorithm 7.1 a simple Noisy Optimization Portfolio Algorithm (NOPA) which does not apply any sharing and distributes the computational budget equally over the noisy optimization solvers.

In this NOPA algorithm, we compare, at iteration r_n, recommendations chosen at iteration $\text{LAG}(r_n)$, and this comparison is based on s_n resamplings, where n is the number of algorithm selection steps. We have designed the algorithm as follows:

Algorithm 7.1 Noisy Optimization Portfolio Algorithm (NOPA).

Input:

noisy optimization solvers $Solver_1, Solver_2, \ldots, Solver_M$

lag function $\text{LAG} : \mathbb{N}^* \mapsto \mathbb{N}^*$ ▷ As in Definition 7.2.4

non-decreasing integer sequence $r_1, r_2, \ldots$ ▷ Periodic comparisons

a non-decreasing integer sequence $s_1, s_2, \ldots$ ▷ Number of resamplings

Output:

approximation $\tilde{x}$ of the optimum x^* of the objective function.

```
n ← 1                                                   ▷ Number of selections
m ← 1                                                   ▷ NOPA's iteration number
i* ← null                                               ▷ Index of recommended solver
x̃ ← null                                                ▷ Recommendation
while not finished do
    if m ≥ r_n then
        i* = argmin_{i∈{1,...,M}} Ê_{s_n}[f(x̃_{i,LAG(r_n)}, ω)]   ▷ Algorithm selection
        n ← n + 1
    else
        for i ∈ {1, ..., M} do
            Apply one evaluation for Solver_i
        end for
        m ← m + 1
    end if
    x̃ = x̃_{i*,m}                                        ▷ Update recommendation
end while
    return x̃
```

- **A stable choice of solver:** The selection algorithm follows the recommendation of the same solver i^* at all iterations in $\{r_n, \ldots, r_{n+1} - 1\}$. This choice is based on comparisons between old recommendations (through the *lag* function LAG).

- **The chosen solver updates are taken into account.** For iteration indices $m < p$ in $\{r_n, \ldots, r_{n+1} - 1\}$, the portfolio chooses the same solver i^*, but does not necessarily recommends the same point because possibly the solver changes its recommendation, i.e., possibly $\tilde{x}_{i^*,m} \neq \tilde{x}_{i^*,p}$.

Effect of the lag. Due to the $\text{LAG}(.)$ function, we compare the $\tilde{x}_{i,\text{LAG}(r_n)}$ (for $i \in \{1, \ldots, M\}$), and not the $\tilde{x}_{i,r_n}$. This is the key point of this algorithm. Comparing the $\tilde{x}_{i,\text{LAG}(r_n)}$ is much cheaper than comparing the $\tilde{x}_{i,r_n}$, because the fitness values are not yet that good at iteration $\text{LAG}(r_n)$, so they can be compared faster - i.e., with less evaluations - than recommendations at iteration r_n. We will make this more formal in Section 7.3, and see under which assumptions this lag has more pros than cons, namely when algorithms have somehow sustained rates. In addition, with lag, we can define INOPA, which saves up significant parts of the computation time.

The first step for formalizing this is to understand the two different kinds of evaluations in portfolio algorithms for noisy optimization. Contrarily to noise-free settings, comparing recommendations requires a dedicated budget, which is far from negligible. It follows that there are two kinds of evaluations:

- **Portfolio budget (Algorithm 7.1, Lines 10-12):** this corresponds to the M evaluations per iteration, dedicated to running the M solvers (one evaluation per solver and per iteration).

- **Comparison budget (Algorithm 7.1, Line 7):** this corresponds to the s_n evaluations per solver at the nth algorithm selection. This is a key difference with deterministic optimization. In deterministic optimization, this budget is zero as the exact fitness value is readily available.

We have $M \cdot r_n$ evaluations in the portfolio budget for the first r_n iterations. We will see below (Section 7.3) conditions under which the other costs (i.e. comparison costs) can be made negligible, whilst preserving the same regret as the best of the M solvers.

INOPA: Improved Noisy Optimization Portfolio Algorithm, with unequal budget

Algorithm 7.2 proposes a variant of NOPA, which distributes the budget in an unfair manner. The solvers with good performance receive a greater budget. The algorithm is designed so that it mimics the behavior of NOPA, but without spending the evaluations which are useless for the moment, given the lag - i.e. we use the fact that evaluations prior to the lagged index are useless except for the selected algorithm.

7.3 Analysis

We here show

- a bound on the performance of NOPA;
- a bound on the performance of INOPA;
- that the *lag* term is necessary.

Preliminary

We define 2 extra properties which are central in the proof.

Definition 7.3.1 ($\mathbf{P_{as}^{(i)}}((\varepsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$). *For any solver $i \in \{1,\dots,M\}$, for some positive sequence $(\varepsilon_n)_{n\in\mathbb{N}^*}$, we define $\mathbf{P_{as}^{(i)}}((\varepsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$:*

$$P_{as}^{(i)}((\varepsilon_n)_{n\in\mathbb{N}^*}) : a.s. \quad \exists n_0,\ \forall n_1 \geq n_0,\ \Delta_{i,n_1} < 2\varepsilon_{n_1} \implies \forall n_2 \geq n_1,\ \Delta_{i,n_2} < 2\varepsilon_{n_2}.$$

Informally speaking, if $\mathbf{P_{as}^{(i)}}((\varepsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ is true, then almost surely for a large enough number of evaluations, the difference between the Simple Regret of solver $i \in \{1,\dots,M\}$ and the optimal Simple Regret is either always at most $2\varepsilon_n$ or always larger - there is no solver infinitely often alternatively at the top level and very weak.

Definition 7.3.2 ($\mathbf{P_{as}}((\varepsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$). *For some positive sequence $(\varepsilon_n)_{n\in\mathbb{N}^*}$, we define $\mathbf{P_{as}}((\varepsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ as follows:*

$$\forall i \in \{1,\dots,M\},\ \mathbf{P_{as}^{(i)}}((\varepsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*}) \text{ holds.}$$

Algorithm 7.2 Improved Noisy Optimization Portfolio Algorithm (INOPA).

Input:

noisy optimization solvers $Solver_1, Solver_2, \ldots, Solver_M$

lag function $\text{LAG} : \mathbb{N}^* \mapsto \mathbb{N}^*$ ▷ As in Definition 7.2.4

non-decreasing integer sequence $r_1, r_2, \ldots$ ▷ Periodic comparisons

a non-decreasing integer sequence $s_1, s_2, \ldots$ ▷ Number of resamplings

Output:

approximation $\tilde{x}$ of the optimum x^* of the objective function.

```
n ← 1                                                   ▷ Number of selections
m ← 1                                                   ▷ INOPA's iteration number
i* ← null                                               ▷ Index of recommended solver
x̃ ← null                                                ▷ Recommendation
while not finished do
    if m ≥ LAG(r_n) or i* = null then
        i* = argmin_{i∈{1,...,M}} Ê_{s_n}[f(x̃_{i,LAG(r_n)}, ω)]   ▷ Algorithm selection
        m' ← r_n
        while m' < r_{n+1} do
            Apply one evaluation to solver i*
            m' ← m' + 1
            x̃ = x̃_{i*,m'}                               ▷ Update recommendation
        end while
        n ← n + 1
    else
        for i ∈ {1,...,M}\i* do
            Apply LAG(r_n) − LAG(r_{n−1}) evaluations for Solver_i
        end for
        m ← m + 1
    end if
end while
    return x̃
```

Remark 7.3.1. *In Definitions 7.3.1 and 7.3.2, we might choose slightly less restrictive definitions, for which the inequalities only hold for integers n such that $\exists i$, $\text{LAG}(r_i) = n$ or $r_i = n$.*

Definitions above can be applied in a very general setting. The Simple Regret of some noisy optimization solvers, for instance Fabian or Polyak-Tsybakov's algorithm, is almost surely $SR_n \leq (1+o(1))\frac{C}{n^\alpha}$ after $n \in \mathbb{N}^*$ evaluations (C is a constant), for some constant $\alpha > 0$ arbitrarily close to 1; see more in Section 2.3, Chap. 2 and Chap. 4 and 5 for Evolution Strategies.

We prove the following proposition for such a case; it will be convenient for illustrating "abstract" general results to standard noisy optimization frameworks.

Property 7.3.1. *Assume that each solver $i \in \{1,\dots,M\}$ has almost surely Simple Regret $(1+o(1))\frac{C_i}{n^{\alpha_i}}$ after $n \in \mathbb{N}^*$ evaluations.*

We define C, α^*, C^*:

$$C = \frac{1}{3}\min\left\{|C_i - C_j| \mid 1 \leq i,j \leq M; C_i - C_j \neq 0\right\}. \tag{7.1}$$

$$\alpha^* = \max_{i\in\{1,\dots,M\}} \alpha_i. \tag{7.2}$$

$$C^* = \min_{i\in\{1,\dots,M\}\ s.t.\ \alpha_i=\alpha^*} C_i. \tag{7.3}$$

We also define the set of optimal solvers:

$$\begin{aligned} SetOptim &= \{i \in \{1,\dots,M\} | \alpha_i = \alpha^*\} \\ \text{and } SubSetOptim &= \{i^* \in SetOptim | C_{i^*} = C^*\} && (7.4) \\ &= \{i \in \{1,\dots,M\} | \alpha_i = \alpha^* \text{ and } C_i = C^*\}. && (7.5) \end{aligned}$$

With these notations, if almost surely, $\forall i \in \{1,\dots,M\}$, the Simple Regret for solver i after $n \in \mathbb{N}^*$ evaluations is $SR_{i,n} = (1+o(1))\frac{C_i}{n^{\alpha_i}}$, then $\mathbf{P_{as}}((\varepsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ is true with ε_n defined as follows:

$$\varepsilon_n = \frac{C}{n^{\alpha^*}}. \tag{7.6}$$

Moreover, if $i_0 \in \{1,\dots,M\}$ satisfies: $(\exists n_0 \in \mathbb{N}^*,\ \forall n \geq n_0,\ \Delta_{i_0,n} \leq 2\varepsilon_n)$, then $i_0 \in SubSetOptim$.

Informally speaking, this means that if the solver i_0 is close, in terms of Simple Regret, to an optimal solver (i.e., a solver matching α^* and C^* in Equations 7.2 and 7.3), then it also has an optimal slope ($\alpha_{i_0} = \alpha^*$) and an optimal constant ($C_{i_0} = C^*$).

Proof. For any solver $i \in \{1,\dots,M\}$ and any solver $i^* \in SubSetOptim$,

$$SR_{i,n} - SR_{i^*,n} = (1+o(1))\frac{C_i}{n^{\alpha_i}} - (1+o(1))\frac{C^*}{n^{\alpha^*}}. \tag{7.7}$$

By Equations 7.6 and 7.7,

$$\frac{SR_{i,n} - SR_{i^*,n}}{\varepsilon_n} = \frac{C_i}{C}\cdot n^{\alpha^*-\alpha_i}\cdot(1+o(1)) - \frac{C^*}{C}\cdot(1+o(1)). \tag{7.8}$$

- If $i \notin SetOptim$, i.e., $\alpha_i < \alpha^*$, the first term in Equation 7.8 tends to ∞, which leads to
$$\lim_{\alpha_i<\alpha^*, n\to\infty} \frac{SR_{i,n} - SR_{i^*,n}}{\varepsilon_n} = \infty.$$
So for all $i \notin SetOptim$, $\exists n_0 \in \mathbb{N}^*$ s.t. $\forall n \geq n_0$, $\Delta_{i,n} = SR_{i,n} - \min_{j\in\{1,\dots,M\}} SR_{j,n} > 2\varepsilon_n$ and, therefore, $\mathbf{P_{as}^{(i)}}((\varepsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ is true.

- If $i \in SetOptim$, i.e., $\alpha_i = \alpha^*$, Equation 7.8 becomes
$$\frac{SR_{i,n} - SR_{i^*,n}}{\varepsilon_n} = \frac{C_i - C^*}{C} + \frac{C_i}{C}o(1) - \frac{C^*}{C}o(1)$$
and therefore
$$\lim_{n\to\infty} \frac{SR_{i,n} - SR_{i^*,n}}{\varepsilon_n} = \frac{C_i - C^*}{C}.$$
  - If $i \in SubSetOptim$, i.e., $C_i = C^*$, $\lim_{n\to\infty} \frac{SR_{i,n}-SR_{i^*,n}}{\varepsilon_n} = 0$. Therefore, $\mathbf{P_{as}^{(i)}}((\varepsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ is true.
  - if $i \notin SubSetOptim$, $\lim_{n\to\infty} \frac{SR_{i,n}-SR_{i^*,n}}{\varepsilon_n} \geq 3$ by definition of C (Equation 7.1). Therefore, $\mathbf{P_{as}^{(i)}}((\varepsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ is true.

So for all $i \in \{1,\dots,M\}$, $\mathbf{P_{as}^{(i)}}((\varepsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ is true, hence $\mathbf{P_{as}}((\varepsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ holds. Moreover, it shows that $\exists n_0 \in \mathbb{N}^*$, $\forall n \geq n_0$, $SR_n^{Solvers} = \min_{j\in\{1,\dots,M\}} SR_{j,n} = SR_{j^*,n}$ where $j^* \in SubSetOptim$.

□

The $\log(M)$-shift for NOPA

We can now enunciate the first main theorem, stating that there is, with fair sharing of the budget as in NOPA, a $\log(M)$-shift, i.e., on a log-log scale (x-axis equal to the number of evaluations and y-axis equal to the log of the Simple Regret), the regret of the portfolio is just shifted by $\log(M)$ on the x-axis.

Theorem 7.3.1 (Regret of NOPA: the $\log(M)$ shift)**.** *Let $(r_n)_{n\in\mathbb{N}^*}$ and $(s_n)_{n\in\mathbb{N}^*}$ be two non-decreasing integer sequences. Assume that:*

- *$\forall x \in \mathscr{D}$, Var $f(x,\omega) \leq 1$;*
- *for some positive sequence $(\varepsilon_n)_{n\in\mathbb{N}^*}$, $\mathbf{P_{as}}((\varepsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ (Definition 7.3.2) is true.*

  Then, there exists n_0 such that:

$$\forall n \geq n_0,\ SR_{r_n}^{Selection} < SR_{r_n}^{Solver} + 2\varepsilon_{r_n} \tag{7.9}$$

$$\textit{with probability at least } 1 - \frac{M}{s_n \varepsilon_{\mathrm{LAG}(r_n)}^2}$$

$$\textit{after } e_n = r_n \cdot M \cdot \left(1 + \sum_{i=1}^{n} \frac{s_i}{r_n}\right) \textit{ evaluations.}$$

Moreover, if (s_n), $\mathrm{LAG}(n)$, (r_n) and (ε_n) satisfy $\sum_{j=1}^{\infty} \frac{1}{s_j \varepsilon_{\mathrm{LAG}(r_j)}^2} < \infty$, then, almost surely, there exists n_0 such that:

$$\forall n \geq n_0,\ SR_{r_n}^{Selection} < SR_{r_n}^{Solver} + 2\varepsilon_{r_n} \tag{7.10}$$

$$\textit{after } e_n = r_n \cdot M \cdot \left(1 + \sum_{i=1}^{n} \frac{s_i}{r_n}\right) \textit{ evaluations.}$$

Remark 7.3.2. *Please notice that Equation 7.9 holds* with a given probability *whereas Equation 7.10 holds* almost surely. *The almost sure convergence in the assumption is proved for some noisy optimization algorithms [Fabian, 1967].*

Proof. First, the total number of evaluations, up to the construction of $\tilde{x}_{i^*_{r_n}, r_n}$ at iteration r_n, is $e_n = M\left(r_n + \sum_{i=1}^{n} s_i\right)$; at this point, each solver has spent r_n evaluations.

Step 1: Proof of Equation 7.9.

By Chebyshev's inequality, for a given $i \in \{1,\dots,M\}$,

$$\mathbb{P}(|\mathbb{E}\left[f(\tilde{x}_{i,\mathrm{LAG}(r_n)},\omega)\right]-\hat{\mathbb{E}}_{s_n}\left[f(\tilde{x}_{i,\mathrm{LAG}(r_n)},\omega)\right]|>\varepsilon_{\mathrm{LAG}(r_n)})<\frac{Var\ f\left(\tilde{x}_{i,\mathrm{LAG}(r_n)}\right)}{s_n\varepsilon^2_{\mathrm{LAG}(r_n)}}\leq\frac{1}{s_n\varepsilon^2_{\mathrm{LAG}(r_n)}}.$$

By union bound,

$$\mathbb{P}(\exists i\in\{1,\dots,M\};|\mathbb{E}\left[f(\tilde{x}_{i,\mathrm{LAG}(r_n)},\omega)\right]-\hat{\mathbb{E}}_{s_n}\left[f(\tilde{x}_{i,\mathrm{LAG}(r_n)},\omega)\right]|>\varepsilon_{\mathrm{LAG}(r_n)})<\frac{M}{s_n\varepsilon^2_{\mathrm{LAG}(r_n)}}.$$

With notation $i^*=i^*_{r_n}=\underset{i\in\{1,\dots,M\}}{\arg\min}\ \hat{\mathbb{E}}_{s_n}\left[f(\tilde{x}_{i,\mathrm{LAG}(r_n)},\omega)\right]$, it follows that, with probability $1-\frac{M}{s_n\varepsilon^2_{\mathrm{LAG}(rn)}}$:

$$\begin{aligned}
\mathbb{E}\left[f(\tilde{x}_{i^*,\mathrm{LAG}(r_n)},\omega)\right] &< \hat{\mathbb{E}}_{s_n}\left[f(\tilde{x}_{i^*,\mathrm{LAG}(r_n)},\omega)\right]+\varepsilon_{\mathrm{LAG}(r_n)};\\
\mathbb{E}\left[f(\tilde{x}_{i^*,\mathrm{LAG}(r_n)},\omega)\right] &< \hat{\mathbb{E}}_{s_n}\left[f(\tilde{x}_{j,\mathrm{LAG}(r_n)},\omega)\right]+\varepsilon_{\mathrm{LAG}(r_n)},\ \forall j\in\{1,\dots,M\};\\
\mathbb{E}\left[f(\tilde{x}_{i^*,\mathrm{LAG}(r_n)},\omega)\right] &< \mathbb{E}\left[f(\tilde{x}_{j,\mathrm{LAG}(r_n)},\omega)\right]+2\varepsilon_{\mathrm{LAG}(r_n)},\ \forall j\in\{1,\dots,M\};\\
\mathbb{E}\left[f(\tilde{x}_{i^*,\mathrm{LAG}(r_n)},\omega)\right]-\mathbb{E}[f(x^*,\omega)] &< \min_{j\in\{1,\dots,M\}} SR_{j,\mathrm{LAG}(r_n)}+2\varepsilon_{\mathrm{LAG}(r_n)};
\end{aligned}$$

So, with probability at least $1-\frac{M}{s_n\varepsilon^2_{\mathrm{LAG}(r_n)}}$,

$$\Delta_{i^*,\mathrm{LAG}(r_n)}<2\varepsilon_{\mathrm{LAG}(r_n)}. \tag{7.11}$$

Using $\mathbf{P_{as}}((\varepsilon_{\mathbf{n}})_{\mathbf{n}\in\mathbb{N}^*})$, Equation 7.11 yields $\Delta_{i^*,r_n}<2\varepsilon_{r_n}$ for $\mathrm{LAG}(r_n)$ large enough, which is the expected result.

Step 2: Proof of Equation 7.10.

We denote by E_n the event "$\Delta_{i^*,r_n}\geq 2\varepsilon_{r_n}$" (equivalent to $SR^{Selection}_{r_n}\geq SR^{Solver}_{r_n}+2\varepsilon_{r_n}$). By Equation 7.9, there exists $n_0\in\mathbb{N}^*$ such that, $\forall n\geq n_0$, $\mathbb{P}(E_n)\leq\frac{M}{s_n\varepsilon^2_{\mathrm{LAG}(r_n)}}$.

Therefore

$$\sum_{j=1}^{\infty}\mathbb{P}(E_j)\leq\sum_{j=1}^{n_0-1}\mathbb{P}(E_j)+M\sum_{j=n_0}^{\infty}\frac{1}{s_j\varepsilon^2_{\mathrm{LAG}(r_j)}}<\infty.$$

According to Borel-Cantelli lemma, almost surely, for n large enough,

$$SR^{Selection}_{r_n}<SR^{Solver}_{r_n}+2\varepsilon_{r_n}$$

and the number of evaluations is still $e_n=r_n\cdot M\cdot\left(1+\sum_{i=1}^{n}\frac{s_i}{r_n}\right)$. □

We now use Proposition 7.3.1 to apply Theorem 7.3.1 on a classical case with almost sure convergence.

Application 7.3.1 ($\log(M)$ shift)**.** *Assume that for any solver $i \in \{1,\dots,M\}$, the Simple Regret after $n \in \mathbb{N}^*$ evaluations is $SR_{i,n} = (1+o(1))\frac{C_i}{n^{\alpha_i}}$. We define $\varepsilon_n = \frac{C}{n^{\alpha^*}}$ (where C and α^* are defined as in Equations 7.1 and 7.2). Assume that $\forall x \in \mathscr{D}$, Var $f(x,\omega) \leq 1$ and that (s_n), $(\text{LAG}(n))$ and (r_n) satisfy:*

$$\begin{aligned} & \textstyle\sum_{j=1}^{\infty} \quad \dfrac{1}{s_j \varepsilon^2_{\text{LAG}(r_j)}} < \infty \\ \textit{and} \quad & \textstyle\sum_{i=1}^{n} \quad s_i = o(r_n). \end{aligned}$$

Then, almost surely,

i) for n large enough, $SR_{r_n}^{Selection} < SR_{r_n}^{Solver} + 2\varepsilon_{r_n}$ after $e_n = r_n \cdot M \cdot \left(1+\sum_{i=1}^{n}\frac{s_i}{r_n}\right)$ function evaluations;

ii) for n large enough, $SR_{r_n}^{Selection} \leq \max\limits_{i \in SubSetOptim} SR_{i,r_n}$ after $e_n = r_n \cdot M \cdot \left(1+\sum_{i=1}^{n}\frac{s_i}{r_n}\right)$ function evaluations;

iii) the slope of the selection regret verifies $\lim\limits_{n\to\infty} \frac{\log(SR_{r_n}^{Selection})}{\log(e_n)} = -\alpha^*$.

$SR_{r_n}^{Selection}$ corresponds to the Simple Regret at iteration r_n of the portfolio, which corresponds to $e_n = r_n \cdot M \cdot \left(1+\sum_{i=1}^{n}\frac{s_i}{r_n}\right)$ evaluations in the portfolio - hence the comment "after e_n function evaluations".

Proof. By Property 7.3.1 and Theorem 7.3.1, i) holds.

By Equation 7.11, and Property 7.3.1, $SR_{r_n}^{Selection} = SR_{i,r_n}$, with $i \in SubSetOptim$ and ii) follows. We obtain:

$$\text{a.s. } \log(SR_{r_n}^{Selection}) = \log(SR_{i,r_n}), \text{ where } i \in SubSetOptim.$$

By the definition of $SubSetOptim$ (Equation 7.5):

$$\begin{aligned} \lim_{n\to\infty} \frac{\log(SR_{r_n}^{Selection})}{\log(e_n)} &= \lim_{n\to\infty} \frac{\log(SR_{i^*,r_n})}{\log(M)+\log(r_n)+\log\left(1+\sum_{i=1}^{n}\frac{s_i}{r_n}\right)} \\ &= \lim_{n\to\infty} \frac{\log(SR_{i^*,r_n})}{\log(r_n)} = -\alpha^*. \end{aligned}$$

Hence iii) holds. □

Example 7.3.1. *The following parametrization matches the conditions in Application 7.3.1.*

$$\begin{aligned} r_n &= \lceil n^{3+r+r'} \rceil; \\ \text{LAG}(n) &= \lceil \log(n) \rceil; \\ s_n &= \lceil n^{1+r'} \rceil,\ r > 0 \text{ and } r' \geq 1,\ n \in \mathbb{N}^*. \end{aligned}$$

The $\log(M')$-shift for INOPA

We now show that INOPA, which distributes the budget in an unfair manner, can have an improvement over NOPA. Instead of a factor M (number of solvers in the portfolio), we get a factor M', number of approximately optimal solvers. This is formalized in the following theorem:

Theorem 7.3.2 ($\log(M')$ shift)**.** *Let $(r_n)_{n\in\mathbb{N}^*}$ and $(s_n)_{n\in\mathbb{N}^*}$ two non-decreasing integer sequences.*

Assume that:

- $\forall x \in \mathscr{D}$, *Var* $f(x,\omega) \leq 1$;
- *for some positive sequence $(\varepsilon_n)_{n\in\mathbb{N}^*}$, $\mathbf{P_{as}}((\varepsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ (Definition 7.3.2) holds.*

We define $S = \{i | \exists n_0 \in \mathbb{N}^*, \forall n \geq n_0, \Delta_{i,n} < 2\varepsilon_n\}$ and M' denotes the cardinality of the set S, i.e., $M' = |S|$. Then, there exists n_0 such that:

$$\forall n \geq n_0,\ SR_{r_n}^{Selection} < SR_{r_n}^{Solver} + 2\varepsilon_{r_n} \tag{7.12}$$

$$\text{with probability at least } 1 - \frac{M}{s_n \varepsilon^2_{\text{LAG}(r_n)}}$$

$$\text{after } e_n = r_n \cdot M' \cdot \left(1 + \frac{M}{M'} \sum_{i=1}^{n} \frac{s_i}{r_n}\right) + (M - M')\text{LAG}(r_n) \text{ evaluations.}$$

Then, if (s_n), $(\text{LAG}(n))$, (r_n) and (ε_n) satisfy $\sum_{j=1}^{\infty} \frac{1}{s_j \varepsilon^2_{\text{LAG}(r_j)}} < \infty$, $\text{LAG}(n) = o(n)$ and $\sum_{j=1}^{n} s_j = o(r_n)$, then, almost surely, there exists n_0 such that:

$$\forall n \geq n_0,\ SR_{r_n}^{Selection} < SR_{r_n}^{Solver} + 2\varepsilon_{r_n} \tag{7.13}$$

$$\text{after } e_n = r_n \cdot M' \cdot (1 + o(1)) \text{ evaluations.}$$

Proof. For a given number of comparisons n, the INOPA algorithm makes the same comparisons and recommends the same value as the NOPA algorithm. Therefore all the results in Theorem 7.3.1 still hold, hence Eqs. 7.12 and 7.13 hold - but we have to prove the number e_n of evaluations.

As the algorithm chooses a solver which is not in S a finite number of times, there exists n_1 such that, for all $n \geq n_1$, the portfolio chooses a solver in S at the nth comparison. We consider $n_0 \geq n_1$ such that $\text{LAG}(n_0) \geq r_{n_1}$. For $n \geq n_0$ the new number of evaluations after n comparisons is:

$$\begin{aligned} e_n &\leq M' \cdot r_n + M \cdot \sum_{i=1}^{n} s_i + (M - M')\text{LAG}(r_n) \\ &= M' \cdot r_n \cdot \left(1 + \frac{M}{M'} \sum_{i=1}^{n} \frac{s_i}{r_n} + \frac{M - M'}{M'} \frac{\text{LAG}(r_n)}{r_n}\right) \\ &= M' \cdot r_n \cdot (1 + o(1)). \end{aligned}$$

□

Using Proposition 7.3.1, we apply Theorem 7.3.2 above to the case of linearly convergent optimization solvers (linear in a log-log scale, with x-axis logarithmic of the number of evaluations and y-axis logarithmic of the Simple Regret).

Application 7.3.2 ($\log(M')$shift)**.** *Assume that $\forall x \in \mathscr{D}$, Var $f(x, \omega) \leq 1$ and for any solver $i \in \{1, \ldots, M\}$, the Simple Regret after $n \in \mathbb{N}^*$ evaluations is $SR_{i,n} = (1 + o(1)) \frac{C_i}{n^{\alpha_i}}$. We define $\varepsilon_n = \frac{C}{n^{\alpha^*}}$ with C and α^* defined as in Eq. 7.1 and 7.2. If $(s_n)_{n \in \mathbb{N}^*}$, $\text{LAG}(n)_{n \in \mathbb{N}^*}$, $(r_n)_{n \in \mathbb{N}^*}$ and $(\varepsilon_n)_{n \in \mathbb{N}^*}$ are chosen such that $\sum_{j=1}^{\infty} \frac{1}{s_j \varepsilon^2_{\text{LAG}(r_j)}} < \infty$, $\text{LAG}(n) = o(n)$ and $\sum_{j=1}^{n} s_j = o(r_n)$, then, almost surely, there exists n_0 such that:*

i) $\forall n \geq n_0$, $SR_{r_n}^{Selection} < SR_{r_n}^{Solver} + 2\varepsilon_{r_n}$ after $e_n = M' \cdot r_n(1 + o(1))$ evaluations;

ii) $\forall n \geq n_0$, $SR_{r_n}^{Selection} \leq \max_{i \in SubSetOptim} SR_{i,r_n}$ after $e_n = M' \cdot r_n(1 + o(1))$ evaluations;

iii) the slope of the selection regret verifies $\lim_{n \to \infty} \frac{\log(SR_{r_n}^{Selection})}{\log(e_n)} = -\alpha^*$.

As usual, $SR_{r_n}^{Selection}$ corresponds to the Simple Regret at iteration r_n of the portfolio, which corresponds to $e_n = r_n \cdot M' \cdot (1 + o(1))$ evaluations in the portfolio - hence the comment "after e_n function evaluations".

Proof. See proof of Application 7.3.1. □

Example 7.3.2. *(log(M') shift) The parametrization of Example 7.3.1 also matches the assumptions of Application 7.3.2.*

The *lag* is necessary

In this section, we show that, if there is no *lag* (i.e., $\forall n,\ \text{LAG}(n) = n$) whenever there are only two solvers, and whenever these solvers have different slopes, the portfolio algorithm might not have a satisfactory behavior, in the sense that, in the example below, it will select infinitely often the worst solver - unless s_n is so large that the comparison budget is not small compared to the portfolio budget.

Example 7.3.3 (The *lag* is necessary)**.** Let us consider the behavior of NOPA without *lag*. We assume the following:

- no lag: $\forall n \in \mathbb{N}^*$, $\text{LAG}(r_n) = r_n$.
- the noise is an additive standard normal distribution $\mathscr{G}$;
- there are $M = 2$ solvers and the two solvers of the portfolio are such that, almost surely, $SR_{i,m} = (1 + o(1))\frac{C_i}{m^{\alpha_i}}$ after $m \in \mathbb{N}^*$ evaluations, $i \in \{1,2\}$, with $\alpha_1 = 1 - e$ and $\alpha_2 = 1 - 2e$, where $e \in [0, 0.5)$ is a constant.
- The comparison budget is moderate compared to the portfolio budget, in the sense that
$$s_n = O(r_n^{\beta}) \tag{7.14}$$
with $\beta \leq 2 - 4e$.

Then, almost surely, the portfolio will select the wrong solver infinitely often.

Proof. Let us assume the scenario above. Let us show that infinitely often, the portfolio will choose the wrong solver. Consider $Y_{1,n}$ and $Y_{2,n}$ defined by

$$Y_{i,n} = \frac{1}{s_n}\sum_{\ell=1}^{s_n} f(\tilde{x}_{i,r_n}, \mathscr{G}^{(i,\ell)}) = \mathbb{E}_{\mathscr{G}}[f(\tilde{x}_{i,r_n}, \mathscr{G})] + Z_i,\ i \in \{1,2\},$$

where

- The $\mathscr{G}^{(i,\ell)}$ are independent Gaussian random variables,
- $Z_i = \frac{1}{s_n}\sum_{\ell=1}^{s_n} \mathscr{G}^{(i,\ell)}$,

- $\tilde{x}_{i,r_n}$ is the search point recommended by solver i after r_n evaluations,

i.e., $Y_{i,n}$ is the average of s_n evaluations of the noisy fitness function at $\tilde{x}_{i,r_n}$.

We denote for all $n \in \mathbb{N}^*$,

$$\delta_n = \mathbb{E}_{\mathscr{G}}[f(\tilde{x}_{2,r_n}, \mathscr{G})] - \mathbb{E}_{\mathscr{G}}[f(\tilde{x}_{1,r_n}, \mathscr{G})] = SR_{2,r_n} - SR_{1,r_n}.$$

δ_n is a random variable, because the expectation operator operates on $\mathscr{G}$; the random dependency in $(\tilde{x}_{1,r_n}, \tilde{x}_{2,r_n})$ remains.

$$v_{1,n} = Var\, Y_{1,n} \text{ and } v_{2,n} = Var\, Y_{2,n}.$$

Definition 7.3.3 ($\mathscr{MR}_n$)**.** *Let $\mathscr{MR}_n$ (misranking at iteration n) be the event "the portfolio chooses the wrong solver at decision step $n \in \mathbb{N}^*$".*

Remark 7.3.3. *From the definitions of solvers 1 and 2, solver 1 is the best in terms of Simple Regret. As a result, if n is big enough, a.s., we get $SR_{1,r_n} < SR_{2,r_n}$, i.e., $\mathbb{E}_{\mathscr{G}}[f(\tilde{x}_{1,r_n}, \mathscr{G})] < \mathbb{E}_{\mathscr{G}}[f(\tilde{x}_{2,r_n}, \mathscr{G})]$. Then it is straightforward that, if a.s. $\mathbb{E}_{\mathscr{G}}[f(\tilde{x}_{2,r_n}, \mathscr{G})] + Z_2 < \mathbb{E}_{\mathscr{G}}[f(\tilde{x}_{1,r_n}, \mathscr{G})] + Z_1$, i.e., $\delta_n < Z_1 - Z_2$, the portfolio chooses solver 2 whereas solver 1 is the best: a.s. $\mathscr{MR}_{r_n}$ occurs.*

Step 1: constructing independent events related to wrong solver choices.

Let us define $\delta'_n = 2(C_2/r_n^{1-2e} - C_1/r_n^{1-e})$. We have

$$\delta'_n = O\left(\frac{C_2}{r_n^{1-2e}}\right) \tag{7.15}$$

Almost surely, $\delta_n = (1+o(1))\frac{C_2}{r_n^{1-2e}} - (1+o(1))\frac{C_1}{r_n^{1-e}}$, for n sufficiently large, $\delta_n < \delta'_n$.

τ_n denotes the event: "$Z_1 - Z_2 > \delta'_n$". So, almost surely, for n sufficiently large, the event $\mathscr{MR}_n$ includes the event τ_n, i.e.

$$\text{almost surely, for } n \text{ sufficiently large, } \tau_n \subset \mathscr{MR}_n. \tag{7.16}$$

Step 2: Almost surely, τ_n occurs infinitely often.

The τ_n are independent, so we apply the converse of Borel-Cantelli lemma. First, compute the probability of τ_n;

$$\begin{aligned} P(\tau_n) &= P\left(\sqrt{v_{1,n}+v_{2,n}}\mathscr{G} > \delta'_n\right), \\ &= P\left(\mathscr{G} > \frac{\delta'_n}{\sqrt{v_{1,n}+v_{2,n}}}\right) \end{aligned}$$

By definition of $v_{1,n}$ and $v_{2,n}$, $\exists C > 0,\ s.t.\ \sqrt{v_{1,n}+v_{2,n}} = \frac{C}{\sqrt{s_n}}$, so by Equation 7.14, $\exists \tilde{C} > 0,\ \frac{1}{\sqrt{v_{1,n}+v_{2,n}}} = \frac{\sqrt{s_n}}{C} \leq \tilde{C} r_n^{\beta/2}$.
By 7.15, $\exists C' > 0\ s.t.\ \frac{\delta'_n}{\sqrt{v_{1,n}+v_{2,n}}} \leq C' \frac{C_2}{r_n^{1-2e}} \tilde{C} r_n^{\beta/2}$, with $\beta/2 = 1-2e$. Hence $P\left(\mathscr{G} > \frac{\delta'_n}{\sqrt{v_{1,n}+v_{2,n}}}\right) \geq P(\mathscr{G} > D)$, with $D > 0$.
We get $P(\tau_n) = \Omega(1)$, as the τ_n are independent, Borel-Cantelli's lemma (converse) implies that almost surely, τ_n occurs infinitely often.

Step 3: Concluding.

Step 2 has shown that almost surely, τ_n occurs infinitely often. Equation 7.16 implies that this is also true for $\mathscr{M}\mathscr{R}_n$.

Therefore, infinitely often, the wrong solver is selected. □

7.4 Conclusion

We have seen that noisy optimization provides a very natural framework for portfolio methods. Different noisy optimization algorithms have extremely different convergence rates (different slopes) on different test cases, depending on the noise level, on the multimodalities, on the dimension. We proposed two versions of such portfolios, NOPA and INOPA, the latter using an unfair distribution of the budget. Both have theoretically the same slope as the best of their solvers, with better constants for INOPA (in particular, no shift, if *SubSetOptim* (see Eq. 7.4) has cardinal 1).

We show mathematically an asymptotic $\log(M)$ shift when using M solvers, when working on a classical log-log scale (classical in noisy optimization); see Section 7.3. Contrarily to noise-free optimization (where a $\log(M)$ shift would be a trivial result), such a shift is not so easily obtained in noisy optimization. Importantly, it is necessary (Section 7.3), for getting the $\log(M)$ shift, that:

- the AS algorithm compares *old* recommendations (and selects a solver from this point of view);
- the portfolio recommends the *current* recommendation of this selected solver.

Additionally, we improve the bound to a $\log(M')$ shift, where M' is the number of optimal solvers, using an unfair distribution of the computational budget (Section 7.3). In particular, the shift is asymptotically negligible when the optimal solver is unique.

A careful choice of portfolio parameters (function $\text{LAG}(\cdot)$, specifying the *lag*; r_n, specifying the intervals $r_{n+1} - r_n$ between two comparisons of solvers; s_n, specifying the number of resamplings of recommendations for selecting the best) leads to such properties; we provide principled tools for choosing these parameters. Sufficient conditions are given in Theorem 7.3.1, with examples thereafter.

Regarding the experiments, we refer to [Cauwet et al., 2014, Cauwet et al., 2016b] for a portfolio of optimization algorithms with the algorithms discussed previously and [Liu and Teytaud, 2014] for multimodal test case. They show (i) the efficiency of portfolios for noisy optimization, as solvers have very different performances for different test cases and NOPA has performance close to the best or even better when the random initialization has a big impact; (ii) the clear and stable improvement provided by INOPA, thanks to an unfair budget distribution; (iii) that the *lag* is usually beneficial, though this is not always the case. Importantly, without lag, INOPA could not be defined.

In noisy frameworks, we point out that portfolios might make sense even when optimizers are not orthogonal. Even with identical solvers, or closely related optimizers, the portfolio can mitigate the effect of unlucky random contributions. This is somehow related to restarts (i.e. multiple runs with random initializations). See [Cauwet et al., 2016b] for cases with very close solvers, and [Liu and Teytaud, 2014] with identical solvers.

Sharing information in portfolios of noisy optimization algorithms is not so easy. Our empirical results are mitigated; but we only tested very simple tools for sharing - just sharing the current best point. A further work consists in identifying better relevant information for sharing; maybe the estimate of the asymptotic fitness value of a solver is the most natural information for sharing; if a fitness value A is already found and a solver claims that it will never do better than A, then we can safely stop its run and save up computational power.

II

Contributions to Delicate Cases

Chapter 8

A Consistent Model Predictive Control

This chapter is based on:

Cauwet, M.-L., Christophe, J.-J., Decock, J., Liu, J., and Teytaud, O. (2016a). A Consistent Model Predictive Control. *To be submitted.*

Disclaimer

Experiments of this chapter have been conducted by Jérémie Decock (Fig, 8.1) and Olivier Teytaud (Fig 8.2) on the top of a Artelys[a] code.

[a] `https://www.artelys.com/en/home`

Various methods, among them Stochastic Dynamic Programming (SDP), Stochastic Dual Dynamic Programming (SDDP), Model Predictive Control (MPC) or Direct Policy Search (DPS) have been developed to tackle the Markov Decision Process (MDP) Problem. We design Direct Model Predictive Control (DMPC)[1], a variant of MPC. Assuming the convergence of the noisy optimization routine, DMPC provably reaches an optimal policy for a wider class of MDP (nonlinear cost and transition functions) than those resolved by MPC (suboptimal by nature), SDP (which needs a moderate size of state space) or SDDP (which

[1] Note that DMPC is also termed Direct Value Search (DVS), e.g. in [Decock, 2014].

requires convexity of Bellman values and a moderate complexity of the random values state) in an acceptable computation time. DMPC clearly outperforms MPC on a multiple-battery management problem, and two hydroelectric problems. The algorithm is easy to implement on top of a classical MPC methodology.

8.1 Introduction

8.1.1 Formalism of Markov Decision Processes

Consider a dynamical system. The system is in a state, and evolves to a new state, depending on random variables and decisions. After a given number of such transitions, the dynamical system is stopped. Each transition provides a cost, and the cost is cumulated from the initial state to this stopping time.

More formally, given an initial state s_0, a policy Π, a transition function $\mathfrak{T}_{nl}$, a random transition function $\mathfrak{RT}$, a final step time T and a sequence of random variables $\omega_0, \ldots, \omega_{T-1} \in \Omega$ (the set of random variables), we define:

$$\forall (s_t, t) \in \mathscr{S} \times \mathscr{D},\ a_t = \Pi(s_t, t) \in L_t(s_t)\text{: the decision at } t \quad (8.1)$$

$$\forall\ (s_t, t, a_t) \in \mathscr{S} \times \mathscr{D} \times L_t(s_t),\ s_{t+1} = \mathfrak{T}_{nl}(s_t, t, a_t) \in \mathscr{S} \quad (8.2)$$

$$\forall (s_t, t, \omega_t) \in \mathscr{S} \times \bar{\mathscr{D}} \times \Omega,\ s_{t+1} = \mathfrak{RT}(s_t, t, \omega_t) \in \mathscr{S} \quad (8.3)$$

$$\forall (s_t, a_t) \in \mathscr{S} \times L_t(s_t),\ c_t = C_{nl,t}(s_t, a_t) \in \mathbb{R}\text{: the cost at time } t \quad (8.4)$$

$$C_\Pi = \sum_{t \in \mathscr{D}} c_t \in \mathbb{R}\text{: the total cost function.} \quad (8.5)$$

Eq. 8.2 refers to decision-based transitions, whereas Eq. 8.3 refers to randomized transitions. $\mathscr{D}$ is the set of time steps at which a decision is made and $\bar{\mathscr{D}}$ is the set of time steps at which we do not make a decision. $\mathscr{S} \subset \mathbb{R}^d$ is the set of states and $L_t(s_t)$ is the set of legal actions in state $s_t \in \mathscr{S}$ at time t. Π is designed to ensure $\Pi(s_t, t) \in L_t(s_t)$.

In all the chapter, we consider:

- a given transition function $\mathfrak{T}_{nl} : \mathscr{S} \times \mathscr{D} \times L_t(s) \to \mathscr{S}$ and a linear approximation $\mathfrak{T}$;
- a given cost function $C_{nl,t} : \mathscr{S} \times L_t(s) \to \mathbb{R}$ and a linear approximation C_t;
- a final step time T;
- an initial state s_0.

Then C_Π is a random variable, only depending on Π. Dynamic optimization is the search for an approximation of Π^* such that $\mathbb{E}[C_{\Pi^*}]$ is as small as possible:

$$\Pi^* \in \arg\min_{\Pi} \mathbb{E}[C_\Pi] \tag{8.6}$$

8.1.2 State of the art in dynamic optimization

We provide an overview of the best known algorithms dedicated to dynamic optimization with continuous state variables and their weaknesses.

Stochastic Dynamic Programming (SDP) goes back to [Bellman, 1957]; it is based on computing, backwards in time, a value function. For a given state, the value function, also termed Bellman function, provides the expected reward that an agent will get, if it starts in this state and makes optimal decisions. Without enhancements, SDP is only tractable for simple problems.

Dual SDP (SDDP), see [Pereira and Pinto, 1991], is the best known improved variant of SDP. Convexity of Bellman values and a random process of moderate state space size [Shapiro, 2011] are required for applying this method. It consists in approximating the value function by linear cuts. Assuming that the Bellman function is convex, a piecewise linear approximation can be obtained thanks to subgradient values at various locations in the state space. SDDP runs simulations and computes subgradients in the visited states. The moderately sized state space of the random process is a key assumption. The Bellman function should be indexed by all state variables, including the variables describing the state of the random process. This is often intractable, hence the random process is often heavily simplified, in particular by stagewise independence [Shapiro, 2011], or at least the random process is replaced by a small Markovian representation.

Model Predictive Control (MPC) is a methodology detailed in [Bertsekas, 2005]. The principle is as follows. In a state s, apply forecasting modules for estimating the random variables of the next h time steps, where h is the tactical horizon. Then, remove all uncertainties - just assume that the forecasts are exact for the next h time steps. Then, make the decision which optimizes the reward over the next h time steps. This methodology is quite simple and the assumptions are clearly understood: we assume that the forecasts are perfect and that the effects beyond the horizon h can be neglected. There are many methods for mitigating these two assumptions. Regarding the first one, it is customary to replace forecasts by pessimistic estimates in order to reduce the risk or to restart the optimization as soon as you get new observations. On the other hand, adding an approximate value function for the reward beyond the

horizon h mitigates the effect of the second assumption. MPC is quite convenient, robust, simple and remains a main tool in spite of theoretical shortcomings. In particular, it often outperforms [Zambelli et al., 2011] the far more sophisticated SDDP which relies on a modelization by a usually stagewise independent random process [Shapiro, 2011] or at least a Markov process with moderate size.

Tree-Based Model Predictive Control (TB-MPC) [Raso, 2013] is an adaptive control method which constructs a tree of possible futures as a forecast and uses this tree to set up a Multi-stage Stochastic Programming (MSP) problem, considering both present uncertainty and its time resolution. The uncertainty will be reduced along the control horizon, and the optimal control strategy (of each branch of tree) after uncertainty reduction will change according to the occurring forecast. TB-MPC therefore also relies on discretization, and has the same shortcomings as MPC for the long term management, but is optimal in the limit of a huge tree-based model.

The principle of Direct Policy Search (DPS) is to optimize a parametric function, to be used as a policy (e.g. [Schoenauer and Ronald, 1994]). The parametric function can be more or less problem specific. Compared to the above approaches, DPS has the strength that it does not need a simplified model; simulation-based optimization can be applied as soon as simulations are possible. On the other hand, traditional DPS fails when confronted with huge action spaces whereas the fact that all above approaches can deal with huge constrained action spaces provided that all involved functions are linear or can be encoded in linear programming, or in other frameworks with moderate complexity. This is rarely the case for DPS, so that except more specialized DPS, it has little relevance to the huge constrained problems from the power systems industry.

8.2 Direct Model Predictive Control (DMPC)

In this chapter, we study Direct Model Predictive Control (DMPC) [Christophe et al., 2014], a variant aimed at bringing consistency into MPC.

8.2.1 DMPC: Formulation

We use Π_θ, the specific policy used in DMPC and parametrized by θ, defined as follows:

$$\forall\ (s_t,t) \in \mathscr{S} \times \mathscr{D},\ \Pi_\theta(s_t,t) \in \underset{a_t \in L_t(s_t)}{\arg\min}\ C_t(s_t,a_t) + \mathscr{N}_\theta[s_t, \mathfrak{T}(s_t,t,a_t),t] \tag{8.7}$$

where $\mathscr{N}_\theta$ is a mapping parametrized by θ, such that for all $(s_t, s'_t, t) \in \mathscr{S} \times \mathscr{S} \times \mathscr{D}$, $\mathscr{N}_\theta[s_t, s'_t, t] \in \mathbb{R}$. In particular, we perform the optimization using the approximations C and $\mathfrak{T}$, and not the 'real' transition and cost functions $C_{nl,.}$ and $\mathfrak{T}_{nl}$ respectively. We will see later how this a priori suboptimality can be mitigated thanks to the valorization term $\mathscr{N}_\theta[.]$.

When $\mathscr{N}_\theta$ is handcrafted rather than optimized, this is MPC. When $\mathscr{N}_\theta$ is computed backwards, this is SDP. When $\mathscr{N}_\theta$ is approximated by linear cuts built on subgradients extracted from simulations, this is SDDP.

In our experiments, $\mathscr{N}_\theta$ is a neural network. The mathematical analysis just assumes that $\mathscr{N}_\theta$ can approximate any necessary function, provided that θ is correctly chosen. This assumption will be formalized in Theorem 8.3.1.

θ is then optimized as follows:

$$\theta^* = \arg\min_\theta C_{\Pi_\theta} \tag{8.8}$$

where C_{Π_θ} is defined by Eqs. 8.1-8.5. C_{Π_θ} is a random variable depending on θ; this is therefore a black-box noisy optimization problem. Eq. 8.8 can be optimized by noisy optimization methods, see Part I.

In addition, we assume that $L_t(s_t)$ is defined by linear constraints, i.e., $\forall (s_t, t) \in \mathscr{S} \times \mathscr{D}$, there exist some matrix $A = A(t, s_t)$ and $B = B(t, s_t)$ such that $L_t(s_t) = \{a_t; As_t \geq a_t \text{ and } Bs_t = b\}$.

8.2.2 DMPC brings consistency into MPC

First and foremost, no assumption on the convexity of the reward is required, contrarily to SDDP and variants of dynamic programming based on convex piecewise linear functions. Also there is no assumption on the Markovian nature of the stochastic processes involved (contrarily to SDDP). Besides, we work in a stochastic setting, contrarily to brute force conformant planning or simple forms of MPC.

We provide a detailed mathematical analysis in Section 8.3 and some large-scale experiments are presented in Section 8.4.

8.3 Consistency analysis

We show that the DMPC structure includes optimal policies under the following linearity assumptions on the approximations $\mathfrak{T}$ and C of $\mathfrak{T}_{nl}$ and $C_{nl,.}$ respectively:

(A1) $\forall (s_t,t) \in \mathscr{S} \times \mathscr{D}$, $a_t \mapsto \mathfrak{T}(s_t,t,a_t)$ is linear;

(A2) $\forall (s_t,t) \in \mathscr{S} \times \mathscr{D}$, $a_t \mapsto C_t(s_t,a_t)$ is linear.

Theorem 8.3.2 states that the DMPC method is optimal under the extra assumptions that the 'real' transition function and cost functions are linear, i.e., $\mathfrak{T} = \mathfrak{T}_{nl}$ and $C_t = C_{nl,t}$ respectively. Then, Section 8.3.2 (Theorem 8.3.3) shows that under an additional technical hypothesis, these extra assumptions can be relaxed.

8.3.1 Optimality of DMPC

We recall that $\|.\|$ denotes the standard Euclidean norm and $\langle .,. \rangle$ denotes the scalar product. In Section 8.3.1, and only in this section, $\mathfrak{T} = \mathfrak{T}_{nl}$ and $C_t = C_{nl,t}$.

Definition 8.3.1. *A mapping $p : \mathscr{S} \times \mathscr{D} \to \mathscr{S}$ will be termed consistent if for all $(s_t,t) \in \mathscr{S} \times \mathscr{D}$, there exists an action $a_t \in L_t(s_t,t)$ such that $\mathfrak{T}(s_t,t,a_t) = p(s_t,t)$.*

Theorem 8.3.1 (DMPC policies can follow arbitrary prescriptions)**.** *Under assumptions (A1) and (A2), for all consistent mapping $p : \mathscr{S} \times \mathscr{D} \to \mathscr{S}$, there exists a mapping $\mathscr{N}_\theta$ such that:*

$$\forall (s_t,t) \in \mathscr{S} \times \mathscr{D},\ \forall\, a \in \underset{a \in L_t(s_t)}{\arg\min}\, C_t(s_t,a) + \mathscr{N}_\theta[s_t, \mathfrak{T}(s_t,t,a), t],\ \mathfrak{T}(s_t,t,a) = p(s_t,t).$$

Furthermore, $\mathscr{N}_\theta[s,.,t]$ can be written as a max of $(d+1)$ linear functions, where d is the dimension of $\mathscr{S}$. The mapping p is termed prescription function.

Proof. Let p be a prescription function. In the following, we use Lemma 8.3.1 (see Appendix D for the proof).

Lemma 8.3.1. *There exist $d+1$ unit vectors $w_1,\dots,w_{d+1}$ in $\mathbb{R}^d$, and there exists a constant $c > 0$ such that, for any unit vector $u \in \mathbb{R}^d$, there exist $i \in \{1,\dots,d+1\}$ such that $\langle u, w_i \rangle > c$.*

We first prove Eqs. 8.9, 8.10 and 8.11.

For all $s \in \mathscr{S}$, we define $v(s) = \max\limits_{i \in \{1,\dots,d+1\}} \langle s, w_i \rangle$, where $(w_i)_{i \in \{1,\dots,d+1\}}$ are as in Lemma 8.3.1. By Lemma 8.3.1, there exists $c > 0$ such that,

$$\forall s \in \mathscr{S} \setminus \{0\}^d, v(s) > c\|s\|. \tag{8.9}$$

Under assumptions (A1) and (A2), by applying [Schrijver, 1986, Theorem 10.5 P126-127], the function $s' \mapsto \min_{a|s'=\mathfrak{T}(s_t,t,a_t)} C_t(s_t,a_t)$ is Lipschitzian, i.e. there exists a constant $D = D(s_t) > 0$ such that $\forall\ (s_1,s_2) \in \mathscr{S} \times \mathscr{S}$,

$$\left| \min_{a_t \in L_t(s_t)|s_1=\mathfrak{T}(s_t,t,a_t)} C_t(s_t,a_t) - \min_{a_t \in L_t(s_t)|s_2=\mathfrak{T}(s_t,t,a_t)} C_t(s_t,a_t) \right| \leq D\|s_1 - s_2\|. \quad (8.10)$$

For a constant $D > 0$ fixed in Eq. 8.10, we define:

$$\forall\ (s,s',t) \in \mathscr{S} \times \mathscr{S} \times \mathscr{D},\ \mathscr{N}_\theta[s,s',t] := \frac{2D}{c} v(s' - p(s,t)). \quad (8.11)$$

Then the minimum of $s' \mapsto \mathscr{N}_\theta[s,s',t]$ is reached in $s' = p(s,t)$ and $\mathscr{N}_\theta[s,.,t]$ is written as a max of $(d+1)$ linear functions.

Define $f(a) = C_t(s_t,a) + \mathscr{N}_\theta[s_t,\mathfrak{T}(s_t,t,a),t]$. Define

$$a^* \in \arg\min_{a_t \in L_t(s_t)|\mathfrak{T}(s_t,t,a_t)=p(s_t,t)} C_t(s_t,a_t) + \mathscr{N}_\theta[s_t,\mathfrak{T}(s_t,t,a_t),t].$$

Such an a^* exists because we have assumed that the prescription is consistent. Then $f(a^*) = C_t(s_t,a^*) + \mathscr{N}_\theta[s_t,p(s_t,t),t] = C_t(s_t,a^*)$.

Let $a' \in L_t(s_t)$ be an action satisfying:

$$a' \in \arg\min_{a_t \in L_t(s_t)} f(a_t)$$

Define $s' = \mathfrak{T}(s_t,t,a')$. To conclude, we need to prove that $s' = p(s_t,t)$. By definition, a' minimize f on $L_t(s_t)$ and a^* minimize f on a subset of $L_t(s_t)$, therefore,

$$\begin{aligned}
f(a^*) &\geq f(a') \\
&\geq C_t(s_t,a') + \mathscr{N}_\theta[s_t,s',t] \\
&\geq f(a^*) - D\|s' - p(s,t)\| + \mathscr{N}_\theta[s_t,s',t] \text{ by Eq. 8.10} \\
&\geq f(a^*) - D\|s' - p(s,t)\| + 2D\|s' - p(s,t)\| \text{ by Eq. 8.11 and Eq. 8.9} \\
&\geq f(a^*) + D\|s' - p(s,t)\|
\end{aligned}$$

Hence $s' = p(s,t)$, which is the expected result. □

Theorem 8.3.2 (The structure of DMPC policies includes optimal policies)**.** *Let us assume (A1) and (A2), and let us assume that $C_t = C_{nl,t}$ and $\mathfrak{T} = \mathfrak{T}_{nl}$. Let us assume that an optimal policy exists*[2]*. Then there exists a mapping $\mathscr{N}_\theta$ such that Π_θ minimizes $\Pi \to \mathbb{E}[C_\Pi]$ i.e. $\forall \Pi, \mathbb{E}[C_\Pi] \geq \mathbb{E}[C_{\Pi_\theta}]$. In addition, $\mathscr{N}_\theta[s,.,t]$ can be written as a max of $(d+1)$ linear functions, where d is the dimension of $\mathscr{S}$.*

Proof. By assumption, there exists Π^* which minimizes $\Pi \to \mathbb{E}[C_\Pi]$.

Let $p^* : \mathscr{S} \times \mathscr{D} \to \mathscr{S}$ be the prescription function defined by:

$$\forall (s_t,t) \in \mathscr{S} \times \mathscr{D},\ p^*(s_t,t) = \mathfrak{T}(s_t,t,\Pi^*(s_t,t)). \tag{8.12}$$

Then, by Theorem 8.3.1, there exists $\mathscr{N}_\theta^*$ such that

$$\begin{aligned} \forall (s_t,t) \in \mathscr{S} \times \mathscr{D}, \forall\, a_t^{dmpc} &\in \underset{a_t \in L_t(s_t)}{\arg\min} C_t(s_t,a_t) + \mathscr{N}_\theta^*[s_t,\mathfrak{T}(s_t,t,a_t),t], \\ \mathfrak{T}(s_t,t,a_t^{dmpc}) &= p^*(s_t,t). \end{aligned} \tag{8.13}$$

Let a_t^* be the action chosen by a given optimal policy Π^* at (s_t,t), i.e. $a_t^* = \Pi^*(s_t,t) \in \underset{\Pi}{\arg\min}\ \mathbb{E}[C_\Pi]$ and by definition, $C_\Pi = \sum_{t \in \mathscr{D}} c_t$. By Eq. 8.12,

$$p^*(s_t,t) = \mathfrak{T}(s_t,t,\Pi^*(s_t,t)) = \mathfrak{T}(s_t,t,a_t^*).$$

Together with Eq. 8.13, $\mathfrak{T}(s_t,t,a_t^{dmpc}) = \mathfrak{T}(s_t,t,a_t^*)$. Then

$$\mathscr{N}_\theta^*[s_t,\mathfrak{T}(s_t,t,a_t^{dmpc}),t] = \mathscr{N}_\theta^*[s_t,\mathfrak{T}(s_t,t,a_t^*),t]. \tag{8.14}$$

Hence, by definition of a_t^{dmpc}, a_t^* and Eq. 8.14, $\forall\ (s_t,t) \in \mathscr{S} \times \mathscr{D}$,

$$\begin{aligned} C_t(s_t,a_t^{dmpc}) + \mathscr{N}_\theta^*[s_t,\mathfrak{T}(s_t,t,a_t^{dmpc}),t] &\leq C_t(s_t,a_t^*) + \mathscr{N}_\theta^*[s_t,\mathfrak{T}(s_t,t,a_t^*),t] \\ C_t(s_t,a_t^{dmpc}) &\leq C_t(s_t,a_t^*). \end{aligned} \tag{8.15}$$

By Theorem 8.3.1, the distribution of $s_0,\ldots,s_T$ is the same with actions chosen by Π^* and with actions chosen by Π_θ. We show this by induction on the step time t:

- The initialization s_0 is the same in both cases.

[2] An optimal policy exists, by Bellman principle of optimality, as soon as relevant extrema in Bellman's equation are reached.

- Induction:

  (I1) By Eq. 8.13, $s_{t+1}|s_t$ is the same in both cases when $t \in \mathscr{D}$.

  (I2) Both are Markov chains, hence $s_{t+1}|s_t$ is the same in both cases when $t \notin \mathscr{D}$.

The expected cost for Bellman's policy Π^* is therefore

$$\mathbb{E} \sum_{t \in \mathscr{D}} C_t(s_t, a_t^*)$$

whereas for DMPC it is

$$\mathbb{E} \sum_{t \in \mathscr{D}} C_t(s_t, a_t^{dmpc}).$$

These two summations are over the same distribution for s_t, and for each s_t, Eq. 8.15 shows that DMPC has a less or equal cost. This concludes the proof of Theorem 8.3.2. □

8.3.2 Nonlinear Setting

In the previous section, Theorem 8.3.2 states that the DMPC methodology provides an asymptotically optimal solution to the Markov Decision Process Problem. To achieve this, it was assumed that the transition and cost functions $\mathfrak{T}_{nl}$ and $C_{nl,.}$ defined in Eqs. 8.2 and 8.4 are linear in the action variable. However, in real word applications, it happens that these assumptions are not satisfied [Frangioni, 2006, Virmani et al., 1989, Najafi and Pourjamal, 2012]. With many algorithms, there is no solution for using nonlinear transition and cost. Usually, to mitigate the computational cost, a linear approximation is used instead of the nonlinear original version. In this section, we want to emphasize that, while a linear approximation is necessary for having a polynomial computational cost, the advantage of simulation-based algorithms (such as DMPC) is that there is no need for *simulating* with the linear functions. Just the function used inside the decision function (Eqs. 8.7 and 8.8) has to be linear. Thanks to the use, *in the simulation part* (Eqs. 8.1 to 8.5), of the real (nonlinear) transition and cost functions, we get an optimal policy on the real problem, in spite of the use (for the sake of polynomial decision time) of linear functions in the policy function Π_θ. We show that DMPC can thereby accommodate such a case under some injectivity condition detailed in Theorem 8.3.3. We can still have an optimal policy if the $\mathscr{N}_\theta$ function is ad hoc. In some sense (made precise in Theorem 8.3.3), the

simulations, performed with the "true" transition $\mathfrak{T}_{nl}$ and cost $C_{nl,.}$, can offset the linear approximations of the nonlinear functions.

Theorem 8.3.3 (DMPC can counterbalance the bias induced by its linear approximations)**.** *Under assumptions (A1) and (A2), and if for some optimal policy* Π^**,*

$$\forall\ (s_t,t) \in S \times \mathscr{D}, \exists!\ a_t \in L_t(s_t), \mathfrak{T}(s_t,t,a_t) = \mathfrak{T}(s_t,t,\Pi^*(s_t,t)) \tag{8.16}$$

where "$\exists!$" stand for "there exists a unique", then for all cost functions $c_0,\dots,c_{T-1}$*, there exists a mapping* $\mathscr{N}_\theta$ *such that the corresponding policy* Π_θ *minimizes* $\Pi \mapsto \mathbb{E}[C_\Pi]$*. Furthermore,* $\mathscr{N}_\theta[s,.,t]$ *can be written as a max of* $(d+1)$ *linear functions, where* d *is the dimension of* $\mathscr{S}$*.*

Proof. Let us define Π^* the optimal policy, for the real transitions and costs (i.e. $\mathfrak{T}_{nl}$ and $C_{nl,t}$, and not their linear counterparts $\mathfrak{T}$ and C_t). The proof of Theorem 8.3.2 applies, except that instead of the prescription $p(s_t,t) = \mathfrak{T}_{nl}(s_t,t,\Pi^*(s_t,t))$ we use the prescription $p(s_t,t) = \mathfrak{T}(s_t,t,\Pi^*(s_t,t))$.

The proof of Theorem 8.3.2 can be applied until Eq. 8.13. Eq. 8.13 and Eq. 8.16 imply that Π_θ plays optimal moves, and therefore it is optimal. □

8.4 Experiments

8.4.1 Experiments with a 10 batteries problem

We first provide experiments on a battery management problem with 10 batteries, as follows: the commands are the power in and power out of each of the 10 batteries; 168 time steps of one hour are considered; operational horizon of 5 hours, tactical horizon of 5 hours; the electricity is sold/bought on a market.

Two algorithms are compared:

- A constant marginal valorization of the stock; we use a battery whenever it saves up more than C euros per MWh. More formally, this means $\mathscr{N}_\theta(s,s',t) = -C\sum_i s'_i$ in Eq. 8.7; s is the stock level of the batteries.
- DMPC, with a penalization (as $\mathscr{N}_\theta$ in Eq. 8.7) provided by a fully connected neural network, with various numbers of neurons on the hidden layer. The inputs to the neural network are the stock levels of the batteries, plus a cosine and a sine function of time with periodicity corresponding to 24 hours.

Results are presented in Fig. 8.1. Results are satisfactory, but need a long optimization time, hence new policies (not a neural network) are designed before switching to large-scale problems in the next section (Section 8.4.2).

8.4.2 Experiments on a real-world hydroelectric problem

Problem description

We here focus on a real world problem, the management of a hydroelectric dam, for national consumption, optimized jointly with many thermal power plants. The naive MPC algorithm does not perform well on this problem, due to the necessary long term management: a valorization term is necessary in the unit commitment, so that water is stored when the stock is low and/or there are still load peaks in the forthcoming months. Two algorithms are compared:

- The MPC improved by a Long Term Management (LTM) constraint on stock level, optimized offline. This is the baseline (horizontal curve in our results), and needs an offline optimization phase (in the present case, combined with human expertise).

- The DMPC approach, with a handcrafted policy rather than a neural network. The handcrafted policy is described in Eq. 8.17:

$$\mathcal{N}_\theta(s,s',t) = (\theta_1 cos(t/T_0) + \theta_2 sin(t/T_0) + \theta_3 + \theta_4 s/S_0) \cdot s'. \tag{8.17}$$

  where t is the time step, T_0 is one year, S_0 is a stock constant, s is a stock level, and $\mathcal{N}_\theta(s,s',t)$ is the valorization as in Eq. 8.7. The two first terms can represent any sinusoid of period $2\pi T_0$, the third is a constant marginal cost of stocks, and the fourth term is a first order approximation of the decrease of marginal costs when a stock increases (law of diminishing returns).

The Long Term Management (LTM) constraint

When applying MPC to a real world problem with long term dependencies (e.g. an hydroelectric stock), it is clearly visible that using a tactical horizon corresponding to the horizon at which forecasts are excellent is not sufficient: costs become huge, because hydroelectric stocks are used way too early. Simulating such a system with 48 hours tactical horizon, without dealing with long term storage, is unrealistic - estimating the cost of a fictitious system with such a simple tool leads to overestimating its cost, in particular if it needs storage.

On the other hand, using a large tactical horizon is unrealistic; we do not have excellent long term forecasts for wind, load and market prices. The cost of a system simulated with one month anticipation is underestimated, in particular

when there are many uncertainties - typically, the cost of a system with a lot of renewable energies is underestimated.

A simple solution is to use human expertise: define a constraint (LTM constraint), lower bound on stock levels, so that you are sure that some energy is saved up in dams. The LTM constraint used in our experiments has been defined by collecting human expertise. This approach leads to MPC costs empirically close to the real world costs; however, this method has the following drawbacks:

- The LTM constraint (extracted by humans by analysis of data) is not effective when the system is modified, and we need to simulate fictitious systems for making investment decisions.
- The one month tactical horizon used in the version designed by human experts is not realistic, as we certainly not have high precision forecasts over one month.
- A pernicious effect is that the LTM constraint is designed over data, and tested on the same data; this is typically an example of *overfitting*. Nothing guarantees that performance will be the same on other data than those used for designing the LTM constraint.

An alternate solution would be optimizing a LTM constraint on top of the MPC. However, this leads to an unprincipled hard constraint on the stock. Indeed, the present work is motivated by the decision to get rid of the LTM constraint and its difficult assumptions.

Experiments on a real-world problem with hydroelectricity

The problem in this section is a nation-wide power system management problem. We consider two frameworks, one with 1 hydroelectric stock and one with 10 hydroelectric stocks are included. Results are presented in Fig. 8.2 (top right, bottom left, bottom right).

We observe that DMPC always outperforms the baselines (horizontal curves). In addition, the results are, to a large extent, independent of the LTM constraint after the optimization by DMPC, whereas the LTM constraint methodology without DMPC had a big impact on the result.

8.5 Conclusion

We have theoretically studied Direct Model Predictive Control. The approach has various advantages and can in particular mitigate the difficulties of high scale dynamic optimization. High scale action spaces can be handled thanks to polynomial decision making, because the neural network provides the parameters of the decision problem but is not involved in the optimization loop (see Eq. 8.7).

Nonlinear transitions can be mitigated by the use of a nonlinear model in simulations (in Eq. 8.8), as shown by Theorem 8.3.3. We can still have an optimal solution. Therefore, we combine polynomial time decision making, and consistency.

Theorem 8.3.2 shows that with a limited number of cuts in the representation (Eq. 8.7) we have approximately optimal policies, outperforming classical MPC. Prior to any optimization, our algorithm is equal to a MPC with, possibly, a handcrafted valorization. It is therefore easy to use, on top of an existing MPC implementation; MPC might be the most efficient usual approach in such contexts, due to its handling of arbitrary complex stochastic processes; we indeed modify it by including a Direct Policy Search layer, so that we preserve the polynomial decision making, while handling nonlinear effects.

The main drawback is the heavy computation of θ^* as in Eq. 8.8. This however can be mitigated by warm starts and parallelization - as noisy optimization is highly parallel. At the cost of a few hours or days of optimization, we get a solution which takes into account nonlinear and stochastic effects. Admittedly, our method is slower than MPC, by far - but we have a principled management of stochastic effects.

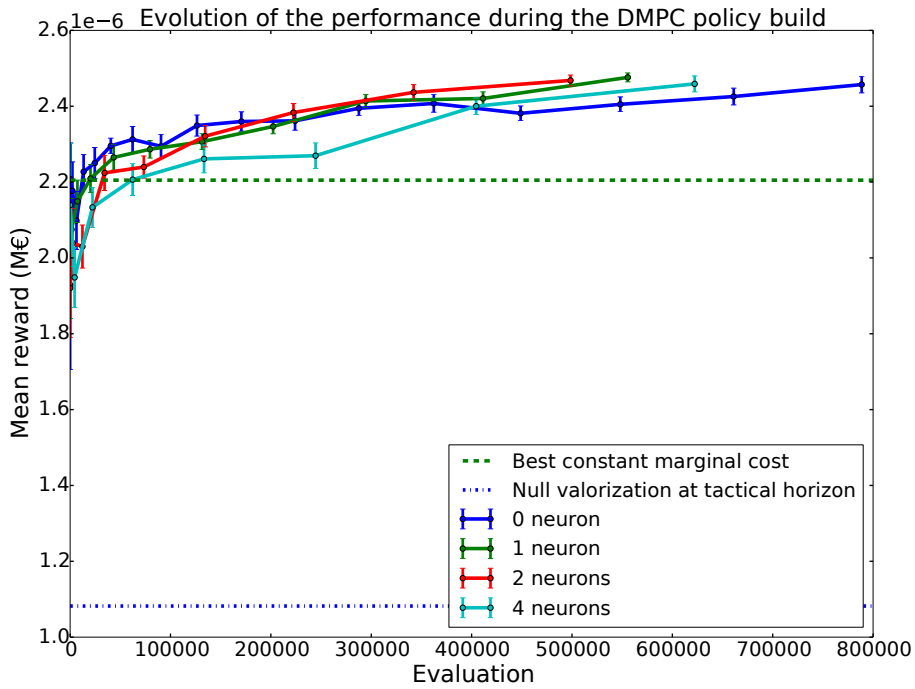


Figure 8.1: Experimental results with 0, 1, 2, 4 neurons over 1000 scenarios (the optimization algorithm is a Self Adaptive Evolution Strategy (4.2.2) with number of resamplings n^2 at iteration n) compared to two baselines: a myopic model (no penalization of stock usage, i.e., Eq. 8.7 with $\mathcal{N}_\theta = 0$) and an optimal constant stock usage, i.e., Eq. 8.7 with $\mathcal{N}_\theta[s_t, \mathfrak{T}(s_t, t, a_t), t] = -C \sum_i \mathfrak{T}(s_t, t, a_t)_i$ for an optimal constant C. 0 neuron is almost immediately better than the baselines; later, 2 neurons perform well; 4 neurons need more computation time for outperforming baselines but eventually works well.

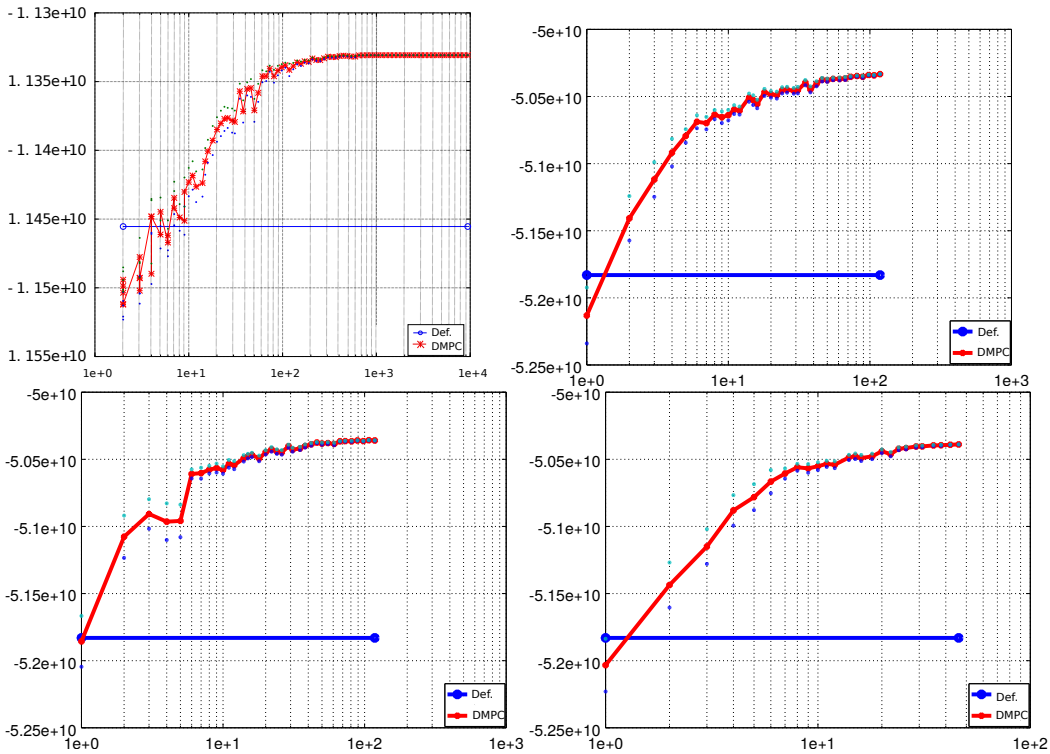


Figure 8.2: Experiments with 1 stock (top left), and 10 stocks (top right: including the LTM constraint designed offline (see Section 8.4.2) by optimization; bottom left: including this constraint on the stocks, but relaxed by 20%; bottom right: constraint completely removed except for the last time step). The horizontal line is, in each case, the performance with the constraint designed offline. The optimization algorithm is a very simple random search, but the structure of the policy is carefully designed (see text). We see that our algorithm has the same result independently of the offline constraint. The X-axis is in all cases the number of evaluations in the optimization of θ. The default methods have no computation of θ; their offline computational costs are larger than time constants here and human expertise is involved in the loop.

Chapter 9

Stochastic zero-sum games

This chapter is based on:

Cauwet, M.-L., St-Pierre, D. L., and Teytaud, O. (2016c). Stochastic zero-sum games. *To be submitted.*

Disclaimer

Experiments of this chapter have been conducted by David Lupien St-Pierre. We have chosen to let them in the present chapter as an illustration of the theoretical part.

This article was initially written within a games theory orientation. We keep here the usual vocabulary and definitions of the games theory. However, as explained in Chap 1 Section 1.2.3, this setting and therefore these results are perfectly relevant in a power system context.

It is known (see [Grigoriadis and Khachiyan, 1995]) that an approximate solution of a Nash equilibrium can be computed in sublinear time with reinforcement learning algorithms. The most efficient algorithm in a parallel setting has been proposed for approximately K processors, considering a $K \times K$ matrix. In this chapter we investigate the extension to stochastic zero-sum games, and the parallelization with more than K processors. First we show the validity of the current state-of-the-art approach (`AdaptedGK`) for the stochastic adversarial case.

Second we propose variants of these algorithms for parallel Nash computation (`ParaGK`). Third we introduce a new algorithm (`ParaNash`) and show both theoretically and empirically its efficiency for approximating a Nash equilibrium in a setting with more than K processors.

9.1 Introduction

The bandit literature is quite rich. For Cumulative Regret in the stochastic setting, [Lai and Robbins, 1985, Auer et al., 2002, Cappé et al., 2013] is close to an optimal solving in the sequential case - though the parallel case remains somehow unexplored. The Simple Regret case was studied in [Bubeck et al., 2011]; basically, it boils down to noisy optimization on finite unstructured domains (i.e. the set of options is finite and has no similarity matrix) with algorithms which are highly parallel - as a consequence, both the sequential and the parallel case are essentially solved. Another research trend is the adversarial case, with [Grigoriadis and Khachiyan, 1995, Auer et al., 1995]. [Auer et al., 1995] has advantages in terms of genericity, but we will here focus on [Grigoriadis and Khachiyan, 1995] because it is provably parallel. This chapter is devoted to this adversarial case and extends results in [Grigoriadis and Khachiyan, 1995] to (i) the stochastic adversarial case (i.e. there is an opponent, and the outcome is stochastic) and (ii) a larger number of processors.

9.1.1 Adversarial Matrix Games & Nash Equilibrium

First, we define adversarial (zero-sum) matrix games. Without loss of generality, throughout the chapter we focus on zero-sum matrix games rather than constant-sum. We remind the definition and properties of the Nash criterion introduced in Chap. 1. Given a $K \times K$ matrix M with coefficients in $[-1,1]$, a game is defined as follows:

- Player 1 chooses (privately) a row $i \in \{1,\dots,K\}$;
- Player 2 chooses a column $j \in \{1,\dots,K\}$ (without observing i);
- Player 1 receives reward $M_{i,j}$ and Player 2 receives reward $1 - M_{i,j}$.

If Player 2 could see i before choosing j, then the game would be trivial: Player 2 would play j minimizing $M_{i,j}$, whereas Player 1 would play (earlier) i

maximizing $\min_j M_{i,j}$. However, here i is not observed by Player 2. In this setting, interesting policies are often not deterministic [Nash, 1952]; Player 1 plays a stochastic policy $p \in [0,1]^K$ ($\sum_{i=1}^K p_i = 1$) and Player 2 plays a stochastic policy $q \in [0,1]^K$ ($\sum_{i=1}^K q_i = 1$). i is actually played by Player 1 with probability p_i and j is (independently) played by Player 2 with probability q_j. The expected payoff for Player 1 is therefore $p^t Mq = \sum_{i,j} p_i M_{i,j} q_j$, and the expected payoff for Player 2 is $1 - p^t Mq$. A Nash equilibrium is a pair (p,q) such that

$$\forall (p',q'),\ p'^t Mq \leq p^t Mq \leq p^t Mq'.$$

Intuitively speaking, at a Nash equilibrium, neither Player 1 nor Player 2 can improve her expected payoff by changing her strategy. It is known that, with K finite, (i) there is always at least one Nash equilibrium; (ii) it is not necessarily unique; (iii) all Nash equilibria (p,q) lead to the same value $v = p^t Mq$. A classical problem is thus the evaluation of a Nash equilibrium, or an approximation thereof.

It is known that an exact Nash equilibrium can be computed in polynomial time [von Stengel, 2002]. However, the degree of this polynomial is large, so that approximate solutions have been proposed. A solution (p,q) is ε-optimal[1] if

$$\forall (p',q'),\ p'^t Mq - \varepsilon \leq p^t Mq \leq p^t Mq' + \varepsilon.$$

In [Grigoriadis and Khachiyan, 1995], a reinforcement learning (bandit style, randomized, with exponential weights) algorithm is proposed for computing an ε-optimal solution in time $O(\log(K)^2/\varepsilon^2)$ on a machine with $K/\log(K)$ processors, with probability $1-\delta$. This is essentially the same complexity $K\log(K)^2/\varepsilon^2$ as `EXP3` [Auer et al., 1995] when applied on a sequential machine (single processor).

In the present chapter, due to the large number of processors available nowadays, we also consider parallel machines with more than K processors. Also, as most of the computation time is usually spent in the evaluation of game results, we consider the number of readings in the matrix instead of time as a complexity measure. Finally, we consider a stochastic setting, in which $M_{i,j}$ is not a real number but a random variable - in this sense, we consider a stochastic adversarial setting.

We adapt existing results to this setting and propose a parallel oracle complexity $\log(K)/\varepsilon$ on a machine with K processors. We also show bounds with K^2 processors and compare experimentally several algorithms in this context.

[1] also termed "ε-approximate Nash equilibrium".

9.1.2 Outline

Section 9.2 introduces the state-of-the-art algorithm, its extension to the stochastic case, its extension to the parallel case with more then K processors, and another proposed algorithm. Section 9.3 analyses theoretically the validity of these algorithms. Section 9.4 shows the experimental results and Section 9.5 concludes.

9.2 Algorithms and settings

In many problems, it makes sense to consider the cost of its request to some black-box oracle instead of the computation time of the algorithm. We justify and explain such setting in Section 9.2.1. Then, Section 9.2.2 presents the state-of-the-art Grigoriadis's algorithm, its extensions to the parallel case with more then K processors and our proposed algorithm.

9.2.1 Settings & motivations

The setting considered in [Grigoriadis and Khachiyan, 1995] is the following:

Setting 1: approximate Nash equilibria of matrix games on parallel machines.

- The input is a $K \times K$ matrix; it is a fixed matrix. Reading $M_{i,j}$ several times provides the same measurement.
- The complexity measure is the computation time on a parallel machine (EREW-PRAM).
- The goal is to output an ε-approximation of a Nash equilibrium, with probability $1-\delta$.

The setting we consider, below, is different for the following reasons:

- A first modification consists in considering an oracle as input. The oracle reads (i, j), and outputs a realization of $M_{i,j}$; several calls to the oracle with a same (i, j) will provide independent random realizations of some unknown random variable $M_{i,j}$.

- A second modification is that we consider as a measure of complexity the number of readings in the matrix M.

This new setting is very relevant when $M_{i,j}$ is the result of a game between strategy i for Player 1 and strategy j for Player 2 and either the game itself, or any of the strategies, is stochastic.

As a summary, we consider the following setting:

Setting 2: approximate Nash equilibria of stochastic oracles on parallel machines.

- The input is a $K \times K$ oracle, which outputs a random realization of $M_{i,j}$ when its input is (i, j).
- The complexity measure is the number of times the oracle is called, given that the oracle can compute C outputs simultaneously, for C inputs $(i_1, j_1), \ldots, (i_C, j_C)$. We refer to this complexity as parallel oracle complexity.
- The goal is to output an ε-approximation of a Nash equilibrium, with probability $1 - \delta$.

This is realistic in various situations:

1. Scenario planning: $M_{i,j}$ is the reward when making the technological decision i and an expensive simulator provides a (randomized) outcome for scenario j. Typically,
   (a) the random part (i.e. the fact that $M_{i,j}$ is a random variable) comes from some random process within the oracle, e.g. when optimizing power systems investments on average over possible weather outcomes.
   (b) the different scenarios are political (e.g. problems in gas supply) and technological (improvements in solar power or electricity storage). An example of scenario-based power systems optimization is [Dong et al., 2011].
2. Choice of opening books in games: $M_{i,j}$ is the reward when choosing opening book i and when the opponent chooses the opening book j. In [Gaudel et al., 2011], $M_{i,j}$ is obtained by simulations with Monte Carlo Tree Search, which is stochastic.

3. Choice of parameters for an agent playing a game: $M_{i,j}$ is the reward when choosing parametrization i and when the opponent chooses parametrization j. This is for example done in [St-Pierre and Teytaud, 2014].

We present algorithms in the case of symmetric games, as in [Grigoriadis and Khachiyan, 1995]. This means that M is a square matrix and $\forall i, j,\ M_{i,j} = -M_{j,i}$.

9.2.2 Algorithms

[Grigoriadis and Khachiyan, 1995] propose a parallel randomized algorithm specific to Setting 1 that we adapt to Setting 2 (labeled `AdaptedGK`) and present in Algorithm 9.1. The key point is to provide a reliable stopping criteria. In the original framework, the algorithm stops when a given variable (namely U_i/t, see Alg. 9.1) is small enough. In Setting 2, this variable is stochastic, hence it does not make any sense to keep the same criteria. We then fix in advance the number of iterations (see Theorem 9.3.1) so that with high probability, the output of the algorithm is an ε-optimal strategy.

We also propose 2 variants of `AdaptedGK` for the parallel case with C processors. The first variant is called `NaiveParaGK`. The idea behind `NaiveParaGK` is to basically resample n times each request to M, with $n = \lceil C/K \rceil$. Algorithm 9.2 presents such a variant. The second variant is labeled `ParaGK`. While `NaiveParaGK` focuses on reading n times the same entry, `ParaGK` chooses simultaneously n columns rather than just one. Thus, it evaluates and updates them all simultaneously (in parallel). Algorithm 9.3 presents this variant.

Moreover we propose `ParaNash` (Algorithm 9.4), a different approach designed for a parallel machine with K^2 processors. The idea is to read each element of the matrix per iteration and update our knowledge of the matrix M accordingly. As for `AdaptedGK` , the stopping criteria is also the key point of this algorithm, see Theorem 9.3.2. In terms of implementation, this algorithm is simple and elegant.

For the sake of readability, the algorithms' pseudocodes are presented at the end of this chapter.

9.3 Theory

We consider Setting 2 (see Section 9.2.1), namely the complexity in terms of the number of requests to the matrix M only. Section 9.3.1 investigates the performance of `AdaptedGK`, the Grigoriadis and Khachiyan's original algorithm adapted

to Setting 2, and shows that we recover existing rates for this more general case. Section 9.3.2 then investigates the performance of `ParaNash`.

9.3.1 Extension of Grigorias and Khachiyan's result to setting 2

As a reminder, the setting is different from [Grigoriadis and Khachiyan, 1995] because in this work we consider $M_{i,j}$ as an oracle, providing realization(s) of a random variable $M_{i,j}$ with expectation $a_{i,j}$, whereas in Grigoriadis' work $M_{i,j}$ are constants. We show that `AdaptedGK` remains valid within such a framework.

Theorem 9.3.1. *Given a symmetric game, for $\delta \in (0,1), \varepsilon > 0$, with probability $1-\delta$, Algorithm 9.1 with parameter $t^* = \left\lceil \frac{3\log(4K/\delta)}{\varepsilon^2} \right\rceil$ provides x such that (x,x) is at most 3ε-optimal.*

Proof. The proof in [Grigoriadis and Khachiyan, 1995] cannot be directly applied to our framework, because they consider a deterministic matrix M. In [Grigoriadis and Khachiyan, 1995], each call to the oracle requesting $M_{i,j}$ gets exactly the same value, whereas we assumed a probability distribution: each call to $M_{i,j}$ returns an independent realization of some random variable, with expectation $a_{i,j} = \mathbb{E}[M_{i,j}]$.

However, the proof in [Grigoriadis and Khachiyan, 1995] can be adapted to this setting, using a slightly adapted algorithm, `AdaptedGK` (Alg. 9.1).

Let $A = (a_{i,k})_{(i,k)\in\{1,\dots,K\}^2}$ be such that $\forall (i,k) \in \{1,\dots,K\}^2$, $\mathbb{E}[M_{i,k}] = a_{i,k}$. The notations follows those of Alg. 9.1 and $\|.\|$ denotes the maximum norm. $P_i(t)$ (resp. $U_i(t)$, $(AX)_i(t)$) denotes the i-th coordinate of vector P (resp. U, AX) at the end of iteration $t \geq 1$.

As the game is symmetric, proving the 3ε-optimality of x with probability $1-\delta$ boils down to prove that $\|Ax\| \leq 3\varepsilon$ with probability $1-\delta$.

Step 1: First we prove that with probability $1-\delta/2$, $\left\|\frac{U(t)}{t} - \frac{AX(t)}{t}\right\| \leq 2\varepsilon$ provided that t is big enough.

Assume that the algorithm selects an index $k \in \{1,\dots,K\}$ at iteration t. First, $\forall i \in \{1,\dots,K\}$, $(U_i(t) - (AX)_i(t))$ is a martingale:

$$\forall i \in \{1,\dots,K\},\ U_i(t) - (AX)_i(t) = U_i(t-1) - (AX)_i(t-1) + (M_{i,k} - a_{i,k}),$$

$$\text{Hence, } \mathbb{E}[U_i(t) - (AX)_i(t) | t-1] = U_i(t-1) - (AX)_i(t-1).$$

Second, for all i and k, $|M_{i,k} - a_{i,k}| \leq 2$. So, Azuma-Hoeffding inequality[2] leads to $\mathbb{P}\left(\left|\frac{U_i(t)-(AX)_i(t)}{t}\right| \geq 2\varepsilon\right) \leq 2\exp\left(\frac{-t\varepsilon^2}{2}\right)$. By union bound, $\left\|\frac{U(t)}{t} - \frac{AX(t)}{t}\right\| \leq 2\varepsilon$ with probability $1-\delta/2$ if $t \geq \frac{2\log(4K/\delta)}{\varepsilon^2}$.

Step 2: Then we prove that with probability $1-\delta/2$, $\left\|\frac{U(t)}{t}\right\| \leq \varepsilon$ if t is conveniently chosen.

From Lines 8 and 11 in Alg. 9.1, for all $i \in \{1,\ldots,K\}$, $P_i(t) = \phi_i(t)/\sum_{j=1}^K \phi_j(t)$ where $\phi_i(t) = \exp(\varepsilon U_i(t)/2)$. The potential function $\Phi(t)$ is defined by: $\Phi(t) = \sum_{i=1}^K \phi_i(t)$. If the algorithm selects the index $k \in \{1,\ldots,K\}$, then:

$$\begin{aligned}
\Phi(t+1) &= \Phi(t)\sum_{i=1}^K P_i(t)\exp\left(\varepsilon M_{i,k}/2\right) \\
\text{and } \mathbb{E}[\Phi(t+1)|\Phi(t)] &= \Phi(t)\sum_{i,k=1}^K P_i(t)P_k(t)\exp\left(\varepsilon M_{i,k}/2\right).
\end{aligned}$$

As $M_{i,k} \in [-1,1]$, $\forall \varepsilon \in (0,1]$, $\exp\left(\varepsilon M_{i,k}/2\right) \leq 1 + \frac{\varepsilon}{2}M_{i,k} + \frac{\varepsilon^2}{6}$. Furthermore, $\sum_{i,k=1}^K P_i(t)P_k(t)a_{i,k} = 0$ (as $a_{i,j} = -a_{j,i}$) and $\sum_{i,k=1}^K P_i(t)P_k(t) = 1$.

As a result, $\mathbb{E}[\Phi(t+1)] \leq \mathbb{E}[\Phi(t)]\left(1+\frac{\varepsilon^2}{6}\right)$, which implies that $\mathbb{E}[\Phi(t)] \leq K\exp\{t\varepsilon^2/6\}$.

Hence, by Markov's inequality, with probability $1-\delta/2$, $\Phi(t) \leq \frac{2K\exp\left(\frac{t\varepsilon^2}{6}\right)}{\delta}$.

Then by definition of $\Phi(t)$, with probability $1-\delta/2$, $\phi_i(t) \leq \frac{2K\exp\left(\frac{t\varepsilon^2}{6}\right)}{\delta}$ for all $i \in \{1,\ldots,K\}$, which leads to:

$$\begin{aligned}
\exp(\varepsilon U_i(t)/2) &\leq \frac{2K\exp\left(\frac{t\varepsilon^2}{6}\right)}{\delta} \\
U_i(t)/t &\leq \frac{2}{\varepsilon t}\log\left(\frac{2K\exp\left(\frac{t\varepsilon^2}{6}\right)}{\delta}\right).
\end{aligned}$$

Then, with probability $1-\delta/2$, $U_i(t)/t \leq \varepsilon$ if $t \geq \frac{3\log(2K/\delta)}{\varepsilon^2}$.
To conclude, by triangular inequality, the result holds as soon as

[2] Azuma-Hoeffding inequality is a generalization of Hoeffding's inequality for martingales.

$t^* \geq \max\left\{\frac{2\log(4K/\delta)}{\varepsilon^2}, \frac{3\log(2K/\delta)}{\varepsilon^2}\right\}$, that is, $t^* = \left\lceil \frac{3\log(4K/\delta)}{\varepsilon^2} \right\rceil$ as stated in the theorem. □

Definition 9.3.1 (Exploitability)**.** *Let (p^*, q^*) be a Nash equilibria and $v = p^* M q^*$ its corresponding value. Then, the exploitability of p, mixed policy for the row player, is $v - \min_{q'} pMq'$ (q' non-negative vector summing to 1). The exploitability of q, mixed policy for the column player, is $\max_{p'} p'Mq - v$ (p' non-negative vector summing to 1). The exploitability of (p,q) is the average of the exploitability of p and of the exploitability of q. In particular, an ε-approximate Nash equilibrium is at most 2ε-exploitable.*

Corollary 9.3.1 (Logarithmic parallel time for `AdaptedGK` with K processors)**.** *If t^* is such that*

$$t^*/\log K \to \infty \text{ as } K \to \infty,$$

and

$$\varepsilon = \sqrt{\frac{3\log(4K/\delta)}{t^*}}, \tag{9.1}$$

then the exploitability of (x,x) proposed by Alg. 9.1 converges weakly to 0 as $K \to \infty$.

Proof. Theorem 9.3.1 shows a bound for the precision ε of the form $O(t^*/\log(K))$, hence the expected result. □

`NaiveParaGK` is similar to Alg. 9.1; it only reduces the variance by averaging over additional resamplings. Therefore the result also applies to this case, using K^2 processors instead of K processors.

9.3.2 Another algorithm with K^2 processors: `ParaNash`

Theorem 9.3.2 (Performance of `ParaNash` with K^2 processors)**.** *Assume $\delta \in (0, \frac{1}{2})$ and $K > 1$. Then with time budget*

$$t^* = \left\lceil \frac{4\log(K) - 2\log(-\frac{11\log(1-\delta)}{24})}{\varepsilon^2} \right\rceil,$$

Alg. 9.4 provides a 2ε-approximated Nash equilibrium (p,q) with probability at least $1-\delta$.

Proof. The notations are those of Algorithm 9.4. Let us assume that

$$t^* \geq \frac{4\log(K) - 2\log(-\frac{11\log(1-\delta)}{24})}{\varepsilon^2}. \tag{9.2}$$

Step 1: A precision ε on the matrix ensures a 2ε-optimality for the approximate Nash.

Define $\varepsilon = \sup_{i,j} |M'_{i,j} - a_{i,j}|$, and assume that (p,q) is the Nash equilibrium for M'.

We prove that (p,q) is at most 2ε-optimal for $A = (a_{i,j})_{(i,j)\in\{1,\dots,K\}^2}$ such that $\mathbb{E}[M_{i,j}] = a_{i,j}$.

Define $f_Z(p,q) = \sum_{1\leq(i,j)\leq K} p_i Z_{i,j} q_j$. Then, for any strategy p',

$$f_A(p,q) \geq f_{M'}(p,q) - \varepsilon \geq f_{M'}(p',q) - \varepsilon \geq f_A(p',q) - 2\varepsilon.$$

Symmetrically, $f_A(p,q) \leq f_A(p,q') + 2\varepsilon$ for any strategy q', which shows that (p,q) is 2ε-optimal.

Therefore, it is sufficient to show that $\sup_{i,j} |M'_{i,j} - a_{i,j}| \leq \varepsilon$.

Step 2: We show that with probability at least $1-\delta$, we have a precision ε on the matrix, i.e. $\sup_{i,j} |M'_{i,j} - a_{i,j}| \leq \varepsilon$.

Define $\delta' = \mathbb{P}(\sup_{i,j} |M'_{i,j} - a_{i,j}| > \varepsilon)$; we have to show that $\delta' \leq \delta$.

Define $d = \sup_{i,j} \mathbb{P}(|M'_{i,j} - a_{i,j}| > \varepsilon)$. Then by independence and union bound

$$\delta' \leq 1 - (1-d)^{K^2}. \tag{9.3}$$

We recall that $M'_{i,j} = \frac{\sum_{k=1}^{t^*} M_{i,j,k}}{t^*}$, where $M_{i,j,k} \in [-1,1]$ is an independent random variable with expectation $a_{i,j}$, $\forall\, k \in \{1,\dots,t^*\}$. By Hoeffding's inequality, we get

$$d \leq 2\exp\left(-\frac{2t^*\varepsilon^2}{2^2}\right). \tag{9.4}$$

By Eq. 9.3 and Eq. 9.4, $\delta' \leq 1 - \left\{1 - 2\exp\left(-2t^*\left(\frac{\varepsilon}{2}\right)^2\right)\right\}^{K^2}$,

$$\begin{aligned}
\left\{1 - 2\exp\left(-2t^*\left(\frac{\varepsilon}{2}\right)^2\right)\right\}^{K^2} &\leq 1 - \delta', \\
\log\left\{1 - 2\exp\left(-2t^*\left(\frac{\varepsilon}{2}\right)^2\right)\right\} &\leq \frac{\log(1-\delta')}{K^2}, \\
1 - 2\exp\left(-2t^*\left(\frac{\varepsilon}{2}\right)^2\right) &\leq \exp\left(\frac{\log(1-\delta')}{K^2}\right).
\end{aligned} \tag{9.5}$$

Using Eq. 9.2,

$$1 - 2\exp\left(-2t^*\left(\frac{\varepsilon}{2}\right)^2\right) \geq 1 + \frac{11\log(1-\delta)}{12K^2}.$$

Combined with Eq. 9.5, this provides

$$\begin{aligned} 1 + \frac{11\log(1-\delta)}{12K^2} &\leq 1 - 2\exp\left(-2t^*\left(\frac{\varepsilon}{2}\right)^2\right) \\ &\leq \exp\left(\frac{\log(1-\delta')}{K^2}\right). \end{aligned} \tag{9.6}$$

We now use, for $x \in [\log(\frac{1}{2})/4, 0]$, $\exp(x) \leq 1 + \frac{11}{12}x$.

$0 < \delta \leq \frac{1}{2}$ and $K > 1$ imply that $x \in [\log(\frac{1}{2})/4, 0]$ with $x = \log(1-\delta)/K^2$. Therefore,

$$\exp\left(\frac{\log(1-\delta)}{K^2}\right) \leq 1 + \frac{11\log(1-\delta)}{12K^2}. \tag{9.7}$$

Eq 9.6 and Eq. 9.7 imply

$$\exp\left(\frac{\log(1-\delta)}{K^2}\right) \leq \exp\left(\frac{\log(1-\delta')}{K^2}\right).$$

which leads to $\delta' \leq \delta$. □

Corollary 9.3.2 (Logarithmic parallel time for `ParaNash` with K^2 processors)**.** *If t^* is defined as a function of K and*

$$t^*/\log K \to \infty \text{ as } K \to \infty,$$

then the exploitability of (p,q) proposed by Alg. 9.4 converges weakly to 0 as $K \to \infty$.

Proof. Theorem 9.3.2 shows a bound for the precision ε of the form $O(t^*/\log(K))$, hence the expected result. □

`ParaNash` does not have a better bound than `NaiveParaGK`. However, we have the same bound in the general adversarial stochastic case, and `ParaNash` has trivially a time bound $O(1)$ for finding the exact solution when the variance vanishes, whereas other algorithms do not. In this sense, we have a better bound in the noise-free case, and we expect better results empirically due to this better behavior in some cases.

9.4 Experiments

In this section we evaluate the performance of approximating a Nash equilibrium with the 4 algorithms introduced previously. The first one is the adapted Grigoriadis and Khachiyan algorithm (`AdaptedGK`), which serves as the baseline; it does not use the additional processors. The second and the third one under study are the variants of Grigoriadis and Khachiyan presented in Section 9.2, namely `NaiveParaGK` and `ParaGK`. Fourth and last is our main algorithm, labeled `ParaNash`.

Note that our experiments are performed with non-symmetric games, which is not a problem as algorithms can be easily adapted to such a setting by a simple reduction presented in [Dantzig, 1963, Grigoriadis and Khachiyan, 1995]; if $\mathscr{M}$ is a matrix $K \times K'$, then we can build the following matrix $\mathscr{M}'$:

$$\mathscr{M}' = \begin{pmatrix} 0 & \mathscr{M} & -e \\ -\mathscr{M}^t & 0 & e \\ e^t & -e^t & 0 \end{pmatrix}$$

with e a $(K \times 1)$-matrix with all coefficients equal to 1 and with ^{t} the transposition operator. This $(K + K' + 1) \times (K + K' + 1)$ matrix verifies $\mathscr{M}'^t_{i,j} = -\mathscr{M}'_{j,i}$ and has values in $[-1, 1]$ if $\mathscr{M}$ has values in $[-1, 1]$. Additionally, an ε-approximate Nash equilibrium of $\mathscr{M}$ can be easily recovered from an ε-approximate Nash equilibrium of $\mathscr{M}'$, as follows:

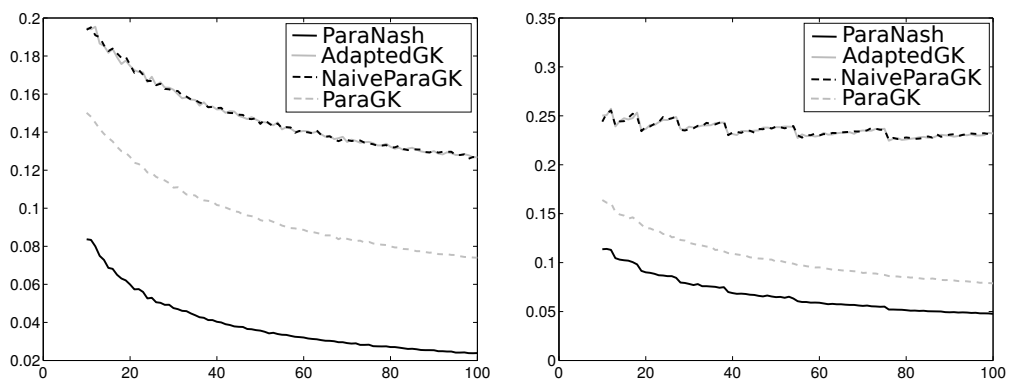
- Assume that (w, w) is a Nash equilibrium of $\mathscr{M}'$; w is a vector of length $K + K' + 1$.
- Then, split $w = (y, x, u)$ where y has length K, x has length K', u is a real number.
- Then, (x, y) is a $O(\varepsilon)$-approximate Nash equilibrium of $\mathscr{M}$, as explained in [Dantzig, 1963, Grigoriadis and Khachiyan, 1995].

Section 9.4.1 evaluates the performance of the algorithms on randomly uniformly drawn Bernoulli parameters, Section 9.4.2 studies other distributions and Section 9.4.3 exhibits some results on the Pokémon problem.

9.4.1 Randomly uniformly drawn Bernoulli parameters

In this section, we consider Bernoulli rewards. At the beginning of each experiment, for each (i, j), $a_{i,j}$ is randomly drawn uniformly in $[0, 1]$; then, for each

(i, j), the reward $M_{i,j}$ when Player 1 plays i and Player 2 plays j is randomly uniformly drawn as a Bernoulli random variable with parameter $a_{i,j}$. The budget is fixed to $t^* = \lceil (\log K)^\alpha \rceil$. The X-axis represents the number of possible choices K and the Y-axis shows the exploitability. Figure 9.1a uses $\alpha = 2.5$ and Figure 9.1b uses $\alpha = 1.5$ which discriminates better among the algorithms. Each experiment is repeated 1 000 times and standard deviations are of the order of magnitude of 10^{-5}. Results are displayed in Figure 9.1.



(a) $\alpha = 2.5$ (b) $\alpha = 1.5$

Figure 9.1: Results for a fixed $t^* = \lceil \log(K)^\alpha \rceil$ with $\alpha = 2.5$ and $\alpha = 1.5$ respectively. X-axis: K. Y-axis: exploitability.

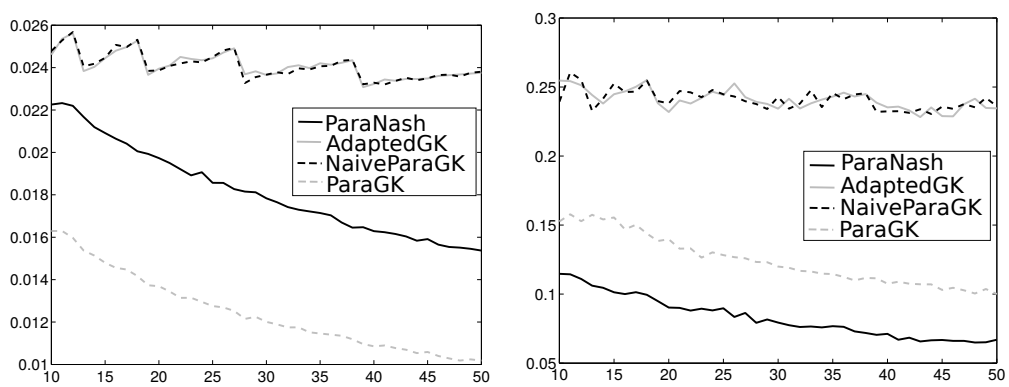
Based on the Corollaries 9.3.1 and 9.3.2, we should observe that each algorithm converges toward 0. We observe in Figure 9.1a that all algorithms have their exploitability converging to 0. There is however a discrepancy in the quality of the performance, with `ParaNash` and `ParaGK` being superior to `AdaptedGK` and `NaiveParaGK`. Thus, it clearly appears that `ParaGK` is the best variant of the Grigoriadis and Khachiyan ones in the parallel case. Moreover, the `ParaNash` algorithm introduced in this paper seemingly outperforms any of the other algorithms when $\alpha > 2$.

We observe in Figure 9.1b that `AdaptedGK` and `NaiveParaGK` seem to converge very slowly toward 0, almost plateauing around 0.25. Again, it appears that `ParaGK` is the best variant of the Grigoriadis and Khachiyan ones in this setting for the parallel case. The `ParaNash` algorithm introduced in this paper seems to outperforms any of the Grigoriadis and Khachiyan variants when $\alpha < 2$.

9.4.2 Other distributions

We provide experiments with different distributions in Figure 9.2. Figure 9.2a is similar to Figure 9.1b, but with Bernoulli parameters uniformly drawn in $[.45, .55]$ rather than $[0,1]$. In Figure 9.2b, for each (i,j), $a_{i,j}$ is uniformly drawn in $[5/12, 7/12]$. Then, rewards $M_{i,j}$ (when Player 1 plays i and Player 2 plays j) are drawn uniformly in $[a_{i,j} - 5/12, a_{i,j} + 5/12]$ at each request to the oracle. The X-axis represents the number of possible choices K and the Y-axis shows the exploitability. In both cases, the budget is fixed to $t^* = \lceil (\log K)^\alpha \rceil$, with $\alpha = 1.5$. Each experiment is repeated 1 000 times and standard deviations are of the order of magnitude of 10^{-5}.

Both Figure 9.2a and 9.2b confirm the good behaviour of `ParaNash` over other problems. In both cases, it clearly appears that `ParaGK` is the best variant in the parallel case, outperforming `ParaNash` in Figure 9.2a. Moreover, we observe similar albeit slower convergence rate to Section 9.4.1. These problems are harder to learn than the previous one (Section 9.4.1) because, for the same budget, their range is smaller. This in turns makes it more difficult to separate based on the observed values between the rows (and columns). This can explain the slower convergence rate.



(a) Uniformly drawn in $[.45, .55]$ (b) Uniformly in $[a_{i,j} - \frac{5}{12}, a_{i,j} + \frac{5}{12}]$

Figure 9.2: Results for a fixed $t^* = \lceil \log(K)^\alpha \rceil$ with $\alpha = 1.5$. X-axis: K. Y-axis: exploitability.

9.4.3 Experiments on the Pokémon problem

The variant of the Pokémon game used in this experiment[3] contains 23 Pokémon, each of them characterized by its type (12 possibilities), its initial and current health points, its attacks (the strength of the attack and the requested energy - 5 energy types - to attack), its weakness (12 possibilities, same as type) and its requested energy to escape. Players 1 and 2 both secretly choose 2 Pokémon among the 23, possibly the same. They choose between 3 actions during the duel: attack, add an energy type or escape. This game can be split into 2 parts: strategic and tactical. The strategic part consists of choosing a set of Pokémon and the tactical part represents the actual combat against an opponent. We used an $\alpha - \beta$ algorithm for the tactical part with a maximum depth of 8 attacks. For the strategic part, we further test our approach by computing a probability distribution over K different combinations of couple of Pokémon using the algorithms presented in this paper. Hence, $M_{i,j}$ is the result (win (+1), loss (-1) or draw(0)) of the duel between deck i (containing 2 Pokémon) and deck j (containing 2 Pokémon as well); $M_{i,j}$ is stochastic, as the same set of Pokémon can lead to different output, depending on the policy followed during the tactical part. We use $\delta = 0.05$, and tested several values of ε for Alg. 9.1, including the one from Corollary 9.3.1. Results are presented in Fig. 9.3 for the ε in Corollary 9.3.1 (Eq. 9.1); manually tuning ε did not provide clear improvement. It clearly appears that the `ParaNash` algorithm again outperforms any of the Grigoriadis and Khachiyan variants.

9.5 Conclusion

We have studied the sequential parallel computation time of approximate Nash equilibria in stochastic adversarial problems. Grigoriadis and Khachiyan, in the adversarial deterministic case, propose an algorithm which efficiently uses K processors (they get a number $K/\log(K)$ of processors with time complexity $\log(K)^2$, but with the parallel oracle complexity they can easily switch to K processors with parallel oracle complexity $\log(K)$).

Adaptation to the stochastic case

In this chapter, we have proposed variants of Grigoriadis & Khachiyan algorithm, adapted to Setting 2 (Section 9.2.1), i.e.: (i) Stochastic setting, namely when $M_{i,j}$

[3] We use the implementation from [Tao Group, 2008]

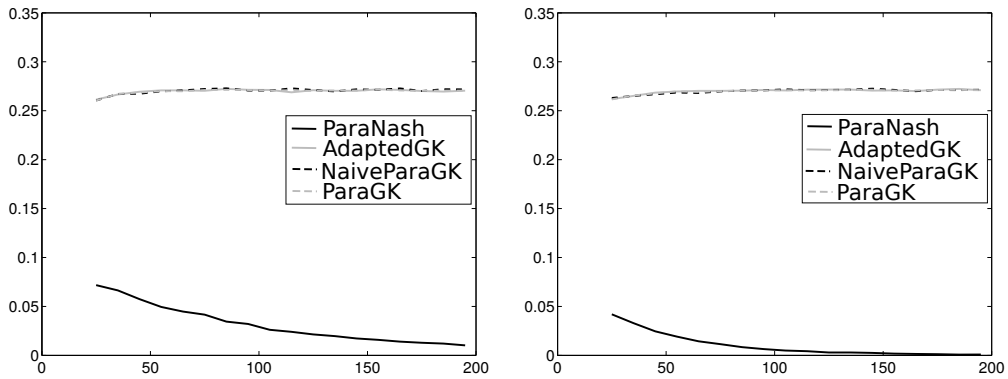


Figure 9.3: Results on the Pokémon game, with $t^* = \lceil \log(K)^\alpha \rceil$ where $\alpha = 2.5$ and $\alpha = 1.5$ respectively. Results are averaged over 1000 runs. X-axis: K. Y-axis: exploitability.

is a randomized oracle. (ii) More than K processors. `AdaptedGK` is our adaptation to Setting 2, using only K processors; we prove (Theorem 9.3.1) that it performs well in such a case, with the same $O(\log(K))$ bound (parallel oracle complexity with K processors) as in the standard case of $M_{i,j}$ deterministic (Setting 1). Corollary 9.3.1 proposes a scaling of ε as a function of t^* and K. It leads to a convergence to the optimum; we therefore get a parameter-free version, where the only parameter is the available computation time and the number K.

Adaptation to K^2 processors

We have considered the case of $C = K^2$ processors. First, we have investigated variants of `AdaptedGK` for such a case. One of our variants, labelled `ParaGK`, outperformed `AdaptedGK` and `NaiveParaGK` by far on all our experiments. We have, however, no proof of the improvement. In terms of theoretical bound, the comparison is as follows.

Stochastic adversarial case. The proved worst case parallel oracle complexity is the same (up to constant factors) with $C = K^2$ processors for `NaiveParaGK` and `ParaNash` (Theorem 9.3.2); and it is the same (within constant factors) as `AdaptedGK` with only K processors. Therefore the parallelization is moderately efficient (in terms of proved bounds) if we have no assumption on the noise.

Deterministic adversarial case. The parallel oracle complexity is $O(1)$ for `ParaNash` in the deterministic case with $C = K^2$ processors (because, in this case, $\forall(i,j),\ M'_{i,j} = a_{i,j}$), whereas `AdaptedGK` still has runtime $\Omega(\log(K))$ in the deterministic case (see tightness in [Grigoriadis and Khachiyan, 1995]).
The additional processors (K^2 instead of K) are therefore (in terms of rates in the proved bounds) useless in the stochastic case (at least, in terms of exponents in the rate) and very efficient in the deterministic case. This chapter did not investigate the intermediate case with low variance. In all our experiments on hard test cases (randomly drawn matrices, with very little structure to be learnt), `ParaNash` and `ParaGK` significantly outperforms the other algorithms. When the variance is large, `ParaGK` performed best; but `ParaNash` is better for other cases, including the real-world one (the Pokémon game).

Further work

An immediate further work is the extension of Theorem 9.3.1 to `ParaGK`, which is a quite intuitive algorithm, performing well in case of small variance, and for which we did not prove anything. `Exp3` [Auer et al., 1995] might also benefit from a parallelization similar to `ParaGK`. Another challenge is the optimality proof in Theorem 9.3.2, i.e. the complexity in the stochastic adversarial case: with K^2 processors, the parallel oracle complexity is $O(\log(K))$ in the general stochastic case but we did not prove tightness. Another further work is the extension of presented results to $C = \lceil K^c \rceil$ processors for $c \geq 1, c \neq 2$.

Algorithm 9.1 Grigoriadis and Khachiyan's algorithm with K processors, where $K \geq 1$, adapted to the case of a stochastic oracle: `AdaptedGK`.

Input:

an accuracy $\varepsilon \in (0,1]$, a risk $\delta \in (0,1)$, $K \geq 1$
an oracle M with K rows and K columns. For $i, j \in \{1, \ldots, K\}^2$, $M_{i,j}$ is an independent output $\in [-1,1]$ with expectation $a_{i,j}$.

Output:

with probability at least $1-\delta$, an ε-optimal strategy x for $\mathbb{E}[M]$ in parallel oracle complexity $O(\varepsilon^{-2}\log K)$

```
t* ← ⌈3log(4K/δ)/ε²⌉
t ← 0
X ← U ← 0 ∈ ℝ^K
∀k ∈ {1,...,K}, P_k ← 1/K
while t < t* do
    t ← t + 1
    Pick a random k ∈ {1,...,K} with probability P_k
    X_k ← X_k + 1
    for i = 1 to K do
        U_i ← U_i + M_{i,k}
    end for
    for i = 1 to K, in parallel do
        P_i ← exp(ε/2 U_i) / Σ_{j=1}^{K} exp(ε/2 U_j)
    end for
end while
    return x = X/t
```

Algorithm 9.2 `NaiveParaGK` with $C \geq K^2$ processors, where $K \geq 1$.

Input:

an accuracy $\varepsilon \in (0,1]$, a risk $\delta \in (0,1)$, $K \geq 1$, $C \geq 1$
an oracle M with K rows and K columns. For $i, j \in \{1, \ldots, K\}^2$, $M_{i,j}$ is an independent output $\in [-1,1]$ with expectation $a_{i,j}$

Output:

with probability $1-\delta$, an ε-optimal strategy x for $\mathbb{E}[M]$ in parallel oracle complexity $O(\varepsilon^{-2}\log K)$

```
t* ← ⌈3log(4K/δ)/ε²⌉
t ← 0
n ← ⌈C/K⌉
X ← U ← 0 ∈ ℝ^K
∀k ∈ {1,...,K}, P_k ← 1/K
while t < t* do
    t ← t + 1
    Pick a random k ∈ {1,...,K} with probability P_k
    X_k ← X_k + 1
    for i = 1 to K do
        U_ik ← 0
        for l = 1 to n in parallel do
            U_ikl ← an independent realization of M_{i,k}
        end for
        for l = 1 to n in parallel do
            U_ik ← U_ik + U_ikl
        end for
        U_i ← U_i + U_ik/n
    end for
    for i = 1 to K in parallel do
        P_i ← exp(ε/2 U_i) / Σ_{j=1}^{K} exp(ε/2 U_j)
    end for
end while
    return x = X/t
```

Algorithm 9.3 `ParaGK` with $C \geq K^2$ processors, where $K \geq 1$.

Input:

an accuracy $\varepsilon \in (0,1]$, a risk $\delta \in (0,1)$, $K \geq 1$, $C \geq 1$
an oracle M with K rows and K columns. For $i, j \in \{1, \ldots, K\}^2$, $M_{i,j}$ is an independent output $\in [-1,1]$ with expectation $a_{i,j}$

Output:

an ε-optimal strategy x for $\mathbb{E}[M]$ in parallel oracle complexity $O(\varepsilon^{-2} \log K)$

```
t* ← ⌈ 3log(4K/δ) / ε² ⌉
t ← 0
n ← ⌈C/K⌉
X ← U ← 0 ∈ ℝ^K
∀k ∈ {1,...,K}, P_k ← 1/K
while t < t* do
    t ← t + 1
    for l = 1 to n in parallel do
        Pick a random k ∈ {1,...,K} with probability P_k
        X_k ← X_k + 1/n
        for i = 1 to K in parallel do
            U_i ← U_i + M_{i,k}
        end for
    end for
    for i = 1 to K in parallel do
        P_i ← exp(ε/2 U_i) / Σ_{j=1}^{K} exp(ε/2 U_j)
    end for
end while
    return x = X/t
```

Algorithm 9.4 `ParaNash` with K^2 processors.

Input:

an accuracy $\varepsilon \in (0,1)$, a risk $\delta \in (0,1/2)$, $K > 1$
an oracle M with K rows and K columns. For $i, j \in \{1,\ldots,K\}^2$, $M_{i,j}$ is an independent output $\in [-1,1]$ with expectation $a_{i,j}$.

Output:

an ε-optimal strategy (p,q) for $\mathbb{E}[M]$ with probability at least $1-\delta$, in parallel oracle complexity $O(\varepsilon^{-2}\log K)$

$t^* \leftarrow \left\lceil \frac{4\log(K) - 2\log(-\frac{11\log(1-\delta)}{24})}{\varepsilon^2} \right\rceil$
for $t = 1$ to t^* **do**
    $\forall i, j \in \{1,\ldots,K\}^2$ sample $M_{i,j}$ in parallel using K^2 processors, get $M_{i,j,t}$
end for
$M' \leftarrow$ the average obtained matrix, i.e. $M'_{i,j} = \frac{1}{t^*}\sum_{t=1}^{t^*} M_{i,j,t}$
$(p,q) \leftarrow$ a Nash equilibrium of M'
    return (p,q)

Chapter 10

Multivariate bias reduction in capacity expansion planning

This chapter is based on:

Cauwet, M.-L. and Teytaud, O. (2016a). Multivariate bias reduction in capacity expansion planning. In *Power Systems Computation Conference, PSCC 2016, Genoa, Italy, June 20-24, 2016*, pages 1–8

To end on the topic of difficult noisy cases, we propose in this chapter a statistical method for handling a finite - and small - data sample. Especially, this is the only part of the present document which does not assume that we have a generative model at hand. We recall in Section 10.1 the setting and challenges of this particular case, introduced in Chap. 1 Section 1.2.3.

10.1 Optimization of power systems capacities: Sample Average Approximation & bias

Quantifying the optimal connection and storage capacities at the scale of a continent or more is a multistage, stochastic and high-dimensional problem. It is multistage due to coupling constraints between time steps, such as stock consistency and warm up costs. Stochasticity comes from the limited precision forecasts: the need for storage and connections varies a lot from one week to another (e.g. power generation is subject to the vagaries of wind) and from one winter to

another (e.g. water inflows due to snow melting). Moreover, the recent years have seen an increase in production volatility, due to the raising use of renewable energies [Pinson, 2013]. It is high-dimensional also since large scale power systems deal with power grid at the continental scale. On the other hand, the sample complexity, i.e. the amount of data necessary for a relevant optimization of capacities, increases linearly with the number of parameters and can be scarcely available at the relevant scale.

Using optimization on average (or risk-aware variants, as well) over a probability distribution is a standard procedure for such problems. The method heavily depends on random processes, modelling weather and consumption. Typically, histories are used, and the objective function is the average cost over this archive of possible weather scenarios [Shapiro, 2011]: such approaches are termed Sample Average Approximation (Section 10.2.1). This method is based on the assumption that the performance of a system can be reliably estimated by checking its performance on the finite set of available past data. They help for fine tuning power systems capacity expansion planning, because they use statistical effects, rather than hard $N-1$ constraints.

We discuss the shortcomings of this approximation (Section 10.2.2) and propose alternate methods. Section 10.3 details some generic resampling tools, then model selection (Section 10.4) is considered in order to mitigate the instability of the resampling method. Section 10.5 presents experimental results.

10.2 M-estimators and bias

In all the chapter, $\hat{\mathbb{E}}_{s\in S}k(s)$ is the average $\frac{1}{n}\sum_{i=1}^{n}k(s_i)$ if S is a sample $S=(s_1,\ldots,s_n)$. $\hat{\mathbb{E}}_{N,\mathfrak{S}}k(s)$ is the empirical average $\frac{1}{N}\sum_{i=1}^{N}k(s_i)$ where $(s_1,\ldots,s_N)$ is a random independently identically drawn sample from $\mathfrak{S}$. $\mathbb{E}_{\mathfrak{S}}$ denotes the expectation over the random process $\mathfrak{S}$. $|\Omega|$ denotes the cardinality of a set Ω.

10.2.1 Sample Average Approximation (SAA)

We denote by $f(s,x)$ the cost when choosing investment x and s is a realization of the random process $\mathfrak{S}$. We want to find x^* such that $\mathbb{E}_{\mathfrak{S}}f(s,x^*)$ is minimal. The *Sample Average Approximation (SAA)* consists in tackling this optimization problem through the use of samples, as the random process is rarely available. We

consider $\hat{x}(S) = \hat{x}$[1] minimizing:

$$x \mapsto \hat{\mathbb{E}}_{s \in S} f(s,x) = \frac{1}{n} \sum_{i=1}^{n} f(s_i, x),$$

with $S = (s_1, \ldots, s_n)$ a sample of independent realizations of the random process $\mathfrak{S}$. Commonly S is an archive. This means that $\hat{x}$ is a M-estimator; it approximates a minimum over a finite sample. When it is obtained by minimizing an empirical estimate as above, it is termed an Empirical Risk Minimizer (ERM) - but not all M-estimators are ERM. For power systems, we need detailed information, which is available for moderate values of n. n is typically between 5 and 100 depending on problems [NERC, 2013, Whitmarsh et al., 2012]. 100 is optimistic as old data is less relevant due to climate change (though corrections are possible), erroneous measurements and missing values.

10.2.2 Bias & Simple Regret

Let us discuss the precision of the SAA in terms of quality of the obtained recommendation. The amount of data samples necessary to estimate properly the parameters of a system increases with the VC-dimension [Vapnik, 1995], which is linear in the number of parameters in smooth cases [Devroye et al., 1997]. The amount of data requested for a given precision is termed the sample complexity. It also increases with the time constants of the problem; if the random processes are only approximately independent when they are 10 years apart from each other, the sample complexity might be multiplied by 10. The sample complexity also increases, typically, quadratically in the inverse precision. Hence, optimizing capacities (both generation capacities and network capacities) against a finite sample can lead to a *bias*. This bias is usually termed overfitting in machine learning [Vaart and Wellner, 1996, Vapnik and Chervonenkis, 1968]. Typically, when SAA is applied, risks are underestimated, and therefore capacities dealing with uncertainties are underestimated, while uncertain assets are overestimated. SAA leads to invest too much in volatile production capacities, but not enough in network and storage capacities.

Let e be an estimate of a quantity x^*, depending on some stochastic random variable $\mathfrak{S}$. Then, the bias b of e is defined by:

$$b = \mathbb{E}_{\mathfrak{S}} e(\mathfrak{S}) - x^*.$$

[1] When needed, the sample S on which the estimate is computed will be specified.

The stochastic random variable $\mathfrak{S}$ is the sample (i.e. S in Section 10.2.1). In many cases, the bias of the ERM estimate $\hat{x}$ defined in Section 10.2.1 is significantly non zero, though it goes to zero asymptotically in the data size [Devroye et al., 1997].

The concept of bias is less widely used in optimization, in particular in the field of large scale power systems, where the huge size of optimization problems makes deterministic optimization already quite hard. However it turns out that, in a renewable energy world, stochasticity really matters [Pinson, 2013]. So this chapter proposes to estimate the bias, and to take it into account in order to improve estimates of x^*.

We define a criterion that measures the quality of an estimate of x^*. The *Simple Regret* of an estimate e is:

$$SR_e = \mathbb{E}_{\mathfrak{S}} f(s,e) - \mathbb{E}_{\mathfrak{S}} f(s,x^*).$$

This is a random variable: the expectation operators operate on s with distribution $\mathfrak{S}$, and SR_e therefore depends only on the (possible) internal randomization of the estimator e. An estimate of the Simple Regret of estimator e will be denoted $\widehat{SR_e}$.

The ERM estimate $\hat{x}$ introduced in Section 10.2.1 is not necessarily the optimal one; the purpose of this chapter is to propose some estimates with Simple Regret less than $SR_{\hat{x}}$. For this, we propose in Section 10.3.1 the use of bias correction tools, namely Jackknife (JK) and Bootstrap (BS). In addition, we reduce the variance of these resampling estimators by a so called dimension reduction method in Section 10.3.2.

10.3 Bias reduction

This section presents resampling estimates, i.e. tools for estimating the bias based on subsamples. Consider a sample $S = (s_1, \ldots, s_n)$ of n realizations of a random process $\mathfrak{S}$. Resampling consists in splitting S into $\hat{S}$ and $\hat{S}'$, usually disjoint. Several such splits could be considered, leading to $\hat{S}_1, \ldots, \hat{S}_N$, and their counterparts $\hat{S}'_1, \ldots, \hat{S}'_N$.

10.3.1 Resampling estimates for bias reduction

Jackknife (JK) or Leave-One-Out (LOO). The Jackknife resampling [Quenouille, 1949], also known as leave-one-out, uses the n subfamilies $\hat{S}_1, \ldots, \hat{S}_n$ of cardinal $n-1$ defined by $\hat{S}_i = (s_1, \ldots, s_{i-1}, s_{i+1}, \ldots, s_n)$. The

complementary family $\hat{S}'_i$ is $\hat{S}'_i = (s_i) = S \setminus \hat{S}_i$. When not all these S_i are required, we can consider a random sample. $\hat{S}$ is then randomly uniformly distributed among $\hat{S}_1, \ldots, \hat{S}_n$ and $\hat{S}'$ is the complementary family. Let us consider $\hat{\hat{x}} = \hat{x}(\hat{S})$ and still $\hat{x} = \hat{x}(S)$. $\hat{\hat{x}}$ depends on $\hat{S}$; it is randomized. This means that in $\hat{\hat{x}}$, we consider the classical M-estimator, but applied to $\hat{S}$ instead of S. The proper bias-corrected estimator for Jackknife $\hat{x}_{JK}$ is [Quenouille, 1949, Efron, 1982]:

$$\hat{x}_{JK} = n\hat{x} - (n-1)\hat{\mathbb{E}}_{N,\hat{S}}\hat{\hat{x}} = n\hat{x} - (n-1)\left[\frac{1}{N}\sum_{j=1}^{N}\hat{x}(\hat{S}_j)\right],$$

where $\hat{S}$ is the random variable defined previously and $(\hat{S}_1, \ldots, \hat{S}_N)$ are N realizations of $\hat{S}$.

Bootstrap (BS). With n the cardinality of the sample S, Bootstrap considers $\hat{S}$ a family of n points, randomly drawn in S, *with replacement*. Without replacement, this would not make any sense, as $\hat{S}$ would be equal to S; but with replacement, it is known that the difference between $\hat{S}$ and S can provide information on the difference between S and the original random process $\mathfrak{S}$ [Efron, 1982]. In Bootstrap, $\hat{S}'$ can also be defined (it is the complementary family of $\hat{S}$ in S); however we do not need it in the present chapter. In the case of Bootstrap, the bias-corrected estimator is:

$$\hat{x}_{BS} = 2\hat{x} - \hat{\mathbb{E}}_{N,\hat{S}}\hat{\hat{x}} = 2\hat{x} - \left[\frac{1}{N}\sum_{j=1}^{N}\hat{x}(\hat{S}_j)\right],$$

where $\hat{S}$ is the random variable defined in the Bootstrap resampling and $(\hat{S}_1, \ldots, \hat{S}_N)$ are N realizations of $\hat{S}$.

$\hat{x}_{JK}$ is usually a better estimate than $\hat{x}_{BS}$, though it is also sometimes mentioned that Bootstrap is more versatile [Wellner, 2014]. We will term these estimates *bias-corrected estimates*. However, the bias, after this correction, is not necessarily zero; it is just, in general, smaller than the bias of $\hat{x}$. The variance, on the other hand, is larger [Scholz, 2007].

10.3.2 Dimension reduction for bias reduction

Let us consider $\hat{x}_r$ a bias-corrected estimate based on resamplings (r stands for resampling), either $\hat{x}_{BS}$ (Bootstrap) or $\hat{x}_{JK}$ (Jackknife). $\hat{x}_r$ has a smaller bias than the original estimate $\hat{x}$, but possibly a larger variance. In high-dimension, $\hat{x}_r$ might be very noisy, and, due to this, we might have $\mathbb{E}SR_{\hat{x}_r} > \mathbb{E}SR_{\hat{x}}$, where the expectation

refers to the random sample S and to the internal randomization of the estimators, including the resampling step.

We define the *absolute dimension reduction* as follows:

$$\hat{x}'_r = \frac{\mu(\hat{x}_r)}{\mu(\hat{x})}\hat{x}, \tag{10.1}$$

with $\mu(v)$ the average of a vector v. This only makes sense if the different capacities have some sort of homogeneity: $(\hat{x})_1,\ldots,(\hat{x})_d$ have the same unit and similar biases, where we denote by $(\hat{x})_i$ the ith component of vector $\hat{x} \in \mathbb{R}^d$.

We also define the *relative dimension reduction* as follows:

$$\hat{x}''_r = \mu(\hat{x}_r/\hat{x})\hat{x}, \tag{10.2}$$

where $u = v/w$ denotes the componentwise division of vector v by vector w, i.e. $(u)_i = (v)_i/(w)_i$ for $i \in \{1,2,\ldots d\}$ if $v \in \mathbb{R}^d$ and $w \in \mathbb{R}^d$.

We will see in our experiments (Section 10.5.3) that the absolute version works better than the relative one.

10.4 Model selection

When several estimates are available, e.g. $\hat{x}_{A_1}(S),\hat{x}_{A_2}(S),\ldots,\hat{x}_{A_k}(S)$, it makes sense to "guess" which one is the best for the data at hand, in order to get a meta-estimate $\hat{x}_{meta}(S)$, which is, depending on some decision rule, equal to $\hat{x}_{A_{i^*}}(S)$ for a $i^* \in \{1,\ldots,k\}$. In our case, model selection can be used for determining which tool, between bias correction methodology and more classical tool such as ERM, should be preferred.

We consider several variants for model selection: classical Cross-Validation (Section 10.4.1) and a recent modification of Cross-Validation, namely Penalized-CV (Section 10.4.2). Using these tools, we combine several estimates into a meta-estimate (Section 10.4.3). The margin method is then proposed for robustification purpose (Section 10.4.4).

As in Section 10.3, we consider a sample $S = (s_1,\ldots,s_n)$ of n realizations of the random process $\mathfrak{S}$, a subfamily $\hat{S}$ of S and its complementary subfamily $\hat{S}' = S \setminus \hat{S}$.

10.4.1 Leave-k-out (Lk) for model selection

Cross-Validation in which the cardinal of $\hat{S}'$ is k is termed leave-k-out (Lk). Lk considers the random variable $\hat{S}$ uniformly distributed among the subfamilies of S of cardinal $n-k$. Leave-one-out is a special case of Cross-Validation, with $k=1$. For model selection, it is classical to use $\hat{S}'$ for testing the estimate built on $\hat{S}$. Given an estimate $e(S)$, depending on a sample S, we define the Lk Cross-Validation error estimate $\hat{\hat{f}}_{\mathrm{L}k}(e)$ by:

$$\begin{aligned}\hat{\hat{f}}_{\mathrm{L}k}(e) &= \hat{\mathbb{E}}_{l,(\hat{S},\hat{S}')}\hat{\mathbb{E}}_{s\in\hat{S}'}f(s,e(\hat{S})) \\ &= \frac{1}{l}\sum_{j=1}^{l}\frac{1}{|\hat{S}'_j|}\sum_{s\in\hat{S}'_j}f(s,e(\hat{S}_j)),\end{aligned}$$

This means that we randomly draw $\hat{S}$ and its complementary family $\hat{S}'$, l times, and each time we build an estimate $e(\hat{S})$ which is tested on $\hat{S}'$. The average result is an estimate of $\mathbb{E}_{\mathfrak{S}}f(s,e(S))$.

For l large enough, increasing k in Lk reduces the variance of $\hat{\hat{f}}_{\mathrm{L}k}(e)$ as an estimate of the real loss $\mathbb{E}_{\mathfrak{S}}f(s,e(S))$, but increases the bias. Penalization (introduced in Section 10.4.2) is a tool for reducing the bias of Cross-Validation, with a moderate increase of the variance.

10.4.2 Penalized-Cross-Validation for model selection (pen$k$-F)

Cross-Validation is classical; we present the Penalized-Cross-Validation method, which, interestingly, is a recent method necessary for making our tool effective in practice.

The Cross-Validation estimates $\hat{\hat{f}}_{\mathrm{L}k}(e)$ is biased since the training data set $\hat{S}$ is smaller than the real data set S. Penalized-Cross-Validation [Arlot, 2008] has been designed to counteract this effect. Informally, it consists in adding a penalization to the current estimated cost. The penalization pen$k$-F is built on S and $\hat{S}$. Given an estimate e, and a sample S, we define:

$$\hat{\hat{f}}_{\text{pen}k\text{-F}}(e) = \hat{\mathbb{E}}_{s\in S} f(s, e(S)) + C \cdot pen(S, e). \tag{10.3}$$

$$\text{with } pen(S,e) = \hat{\mathbb{E}}_{l,\hat{S}} \left[\hat{\mathbb{E}}_{s\in S} f(s, e(\hat{S})) - \hat{\mathbb{E}}_{s\in \hat{S}} f(s, e(\hat{S})) \right],$$

$$= \frac{1}{l} \sum_{j=1}^{l} \left[\frac{1}{|S|} \sum_{s\in S} f(s, e(\hat{S}_j)) - \frac{1}{|\hat{S}_j|} \sum_{s\in \hat{S}_j} f(s, e(\hat{S}_j)) \right].$$

$\hat{S}$ is a random variable as in the Cross-Validation section and C is an overpenalization constant. In other words, we randomly draw l times a subsample (of size $n-k$) $\hat{S}$ from S, each time we build an estimate $e(\hat{S})$ which is tested both on S and $\hat{S}$. The average difference between these costs is the penalization, and the estimate of $\mathbb{E}_{\mathfrak{S}} f(s, e(S))$ is given by Equation 10.3; it is provably optimal in some simple cases [Arlot, 2008].

10.4.3 Meta-estimate using model selection

Typically, we will define for example

$$\hat{x}_{meta,\text{L}k}(\hat{x}, \hat{x}_{JK}) = \begin{cases} \hat{x} & \text{if } \hat{\hat{f}}_{\text{L}k}(\hat{x}) < \hat{\hat{f}}_{\text{L}k}(\hat{x}_{JK}) \text{ (i.e. } \widehat{SR_{\hat{x}}} < \widehat{SR_{\hat{x}_{JK}}}) \\ \hat{x}_{JK} & \text{otherwise} \end{cases}$$

We use the Jackknife-corrected estimator if it is seemingly better (i.e. with a smaller estimate Simple Regret) than the simple M-estimator for the Leave-k-out Cross-Validation. More generally, given k estimators $\hat{x}_{A_1}, \ldots, \hat{x}_{A_k}$, and a model selection MS, $\hat{x}_{meta,MS}(\hat{x}_{A_1}, \ldots, \hat{x}_{A_k})$ is equal to the estimate $\hat{x}_{A_{i^*}}$ which is considered the best by the model-selection MS, i.e. such that $\hat{\hat{f}}_{MS}(\hat{x}_{A_{i^*}})$ is minimum.

10.4.4 The margin method

Let us consider the case in which we have k estimators $(\hat{x}_{A_1}, \ldots, \hat{x}_{A_k})$. Let us assume that A_1 is the default solution (ERM, in our case), that we wish to outperform with our new estimate. We use Penalized-Cross-Validation for selecting one of them. Let us call $\hat{x}_{meta}$ the resulting estimate, equal to $\hat{x}_{A_{i^*}}$, for a $i^* \in \{1, \ldots, k\}$, depending on the Penalized-Cross-Validation results. The result is satisfactory in most cases, but there are test cases in which $\hat{x}_{A_1}$ is better than $\hat{x}_{A_{i^*}}$ with $1 \neq i^*$,

because the Penalized-Cross-Validation fails in finding the best among the k estimates. Then, we propose the following method, termed margin method: instead of comparing the estimated Simple Regrets $(\widehat{SR_{A_1}}, \ldots, \widehat{SR_{A_k}})$, of $(\hat{x}_{A_1}, \ldots, \hat{x}_{A_k})$ respectively, compare $((\mathbf{1}-\gamma)\widehat{SR_{A_1}}, \widehat{SR_{A_2}}, \ldots, \widehat{SR_{A_k}})$, for some $\gamma \in (0,1)$. Then, we expect the estimator $\hat{x}_{meta}$ to be more robust, in the sense that it is rarely worse than the original $\hat{x}_{A_1}$ - the $(k-1)$ others estimates $(\hat{x}_{A_2}, \ldots, \hat{x}_{A_k})$ are used only if the model selection considers $\hat{x}_{A_1}$ to be outperformed by far.

10.5 Experiments

Section 10.5.1 presents the experimentation framework, Section 10.5.2 lists the estimators considered and Section 10.5.3 presents the experimental results.

10.5.1 Test case

Let us consider an electric grid, connected to d distinct areas. An area $i \in \{1, \ldots, d\}$ is connected to the main grid only through one connection, with capacity $x^{(i)}$. The connection must be large enough so that the flow does not exceed the capacity, but larger connections are more expensive. Hence we should find a good compromise.

The cost function, when the maximum consumption over the year is $s = \left(s^{(1)}, \ldots, s^{(d)}\right)$, for a non-negative $x = \left(x^{(1)}, \ldots, x^{(d)}\right)$, is

$$f(s,x) = p \times \left(\sum_{i=1}^{d} \mathbf{1}_{s^{(i)} > x^{(i)}} \right) + \sum_{i=1}^{d} x^{(i)}, \tag{10.4}$$

where

- p is a parameter: it is the penalty in case of fault, compared to the cost of 1 unit of network capacity.
- $x^{(i)}$ is the ith network capacity, i.e. the capacity connecting area number i to the main grid.

Faults have long lasting consequence, far beyond the time during which the flow exceeds the capacities; hence the "binary" nature of the penalization. It is a common practice in power systems [NERC, 2013, Whitmarsh et al., 2012] to consider the maximum over the year, and not the number of times or number of hours an

overflow occurs. This is because overflows can lead to various problems, with an impact lasting long after the overflow itself, thus it does not make sense to consider that an overflow is less important just because it is short or occurred just once.

For this artificial experiment, the random process s is a discrete distribution (with support of cardinal 1500) generated as follows: *cov* standard centered Gaussian random variables are independently randomly drawn, in dimension d, with *cov* an integer. Let us define their covariance by V; hence, V is the identity if $cov \to \infty$ but might be far from identity when *cov* is small. s (resp. $\log(s)$ in the heavily-tailed case) is the d dimensional centered Gaussian with covariance V.

The greater *cov*, the simpler the problem. Roughly speaking, *cov* large makes all areas $i \in \{1, \ldots, d\}$ more similar. Here, log refers to the logarithm with natural basis. $\log(x)$, when x is a d-dimensional vector, refers to $(\log(x^{(1)}), \ldots, \log(x^{(d)}))$.

10.5.2 Estimators

We compare the estimators of Table 10.1. For the *meta* versions (section 10.4.3), we also test the version with the penalization method (Section 10.4.2) and with the "margin" method (Section 10.4.4).

10.5.3 Experimental results

Results with Bias Reduction and Model Selection only. In this section, resamplings for model selection are based on $\#samples/k$ (see Figs. 10.1, 10.2 and 10.3) random splits of the data into Cross-Validation folds of size k. Supplementary experimental results are presented in the extended material (`https://www.lri.fr/~cauwet/liste.pdf`). We here provide a sample of results.

Figs. 10.1, 10.2 and 10.3 present the Simple Regret(averaged over 200 independent runs) of the various estimators in function of the sample size. **god** refers to the optimal solution; it has regret 0, by definition. **ERM** denotes the classical M-estimator $\hat{x}$. **resample** refers to an estimate with bias reduction: $\hat{x}_{BS}$ or $\hat{x}_{JK}$. **sideRS** (resp. **sideRS2**) refers to $\hat{x}'_{BS}$ or $\hat{x}'_{JK}$ (resp. $\hat{x}''_{BS}$ or $\hat{x}''_{JK}$). **CV** refers to $\hat{x}_{meta,L1}$ (top left and right) or $\hat{x}_{meta,L3}$ (bottom left and right). For each subfigure: on the left (both top and bottom) **resample**, **sideRS** and **sideRS2** are performed with Bootstrap; on the right (both top and bottom) **resample**, **sideRS** and **sideRS2** are performed with Jackknife.

In Fig. 10.1 there is a strong bias in the M-estimator, due to the strong penalty (the loss function has a strong third derivative at the optimum). In this case:

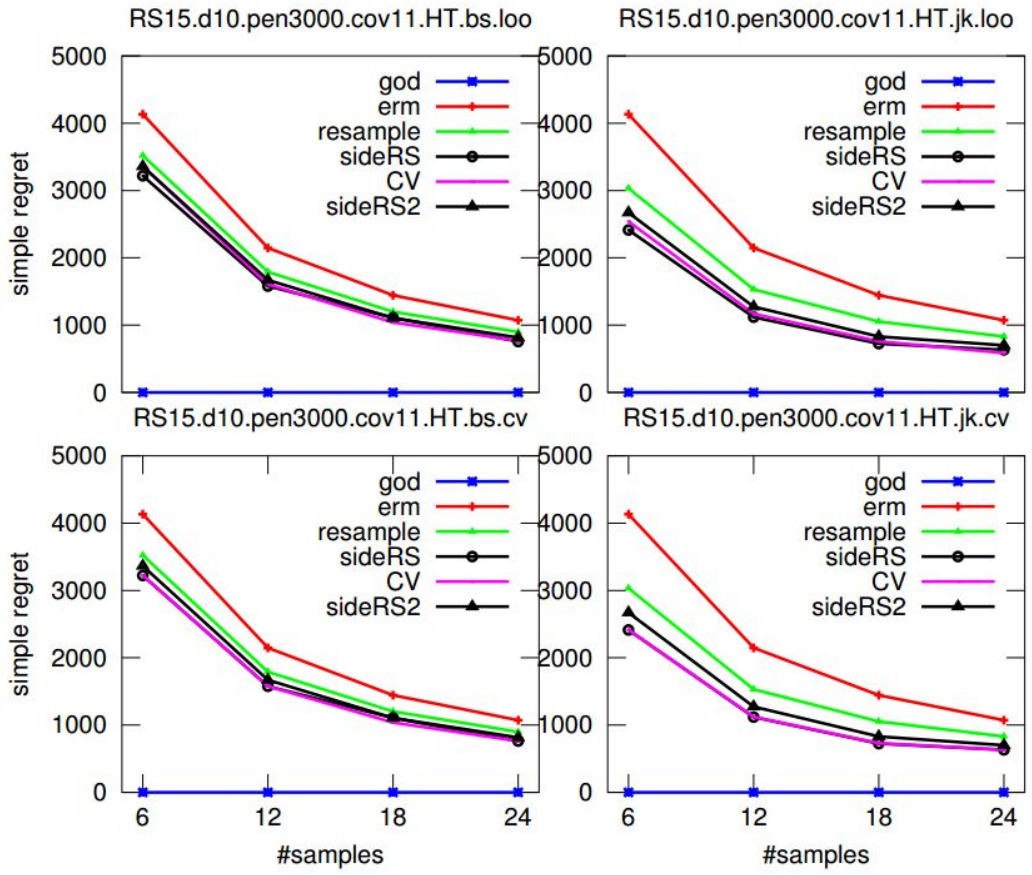


Figure 10.1: $d = 10$, $cov = 11$, $p = 3000$, 15 resamplings for the bias corrections, heavy tail.

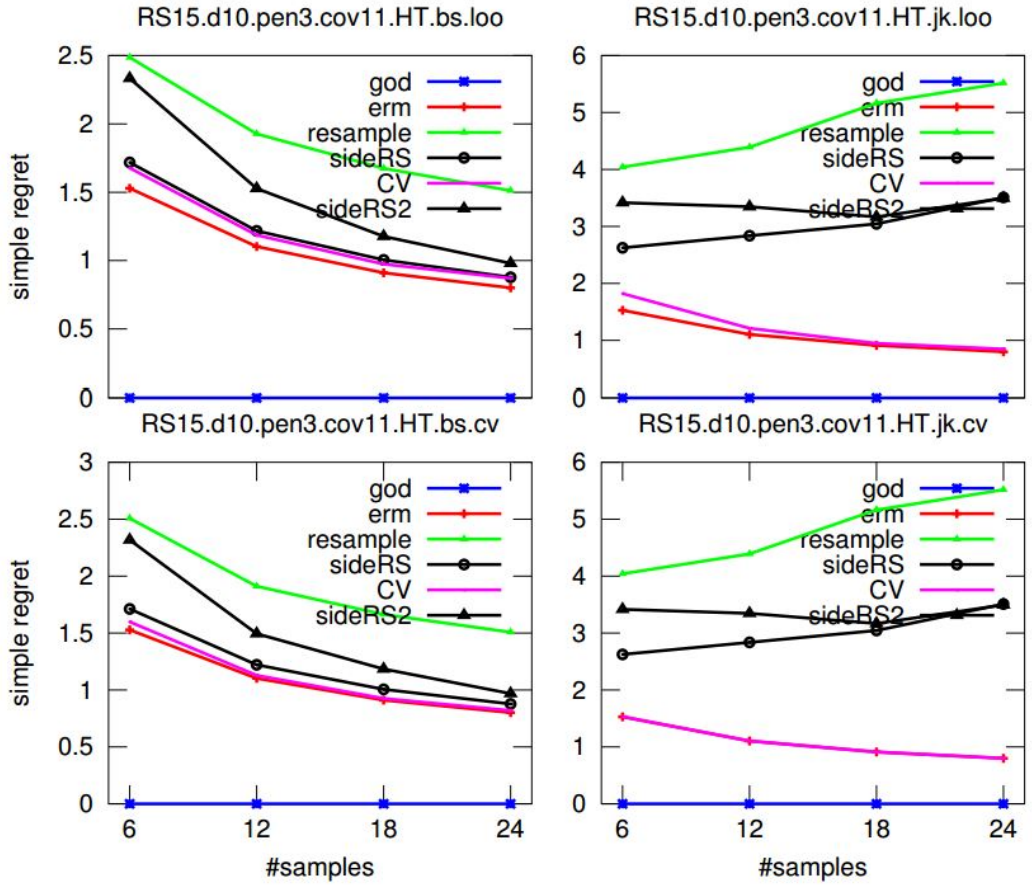


Figure 10.2: $d = 10$, $cov = 11$, $p = 3$, 15 resamplings for the bias correction, heavy tail.

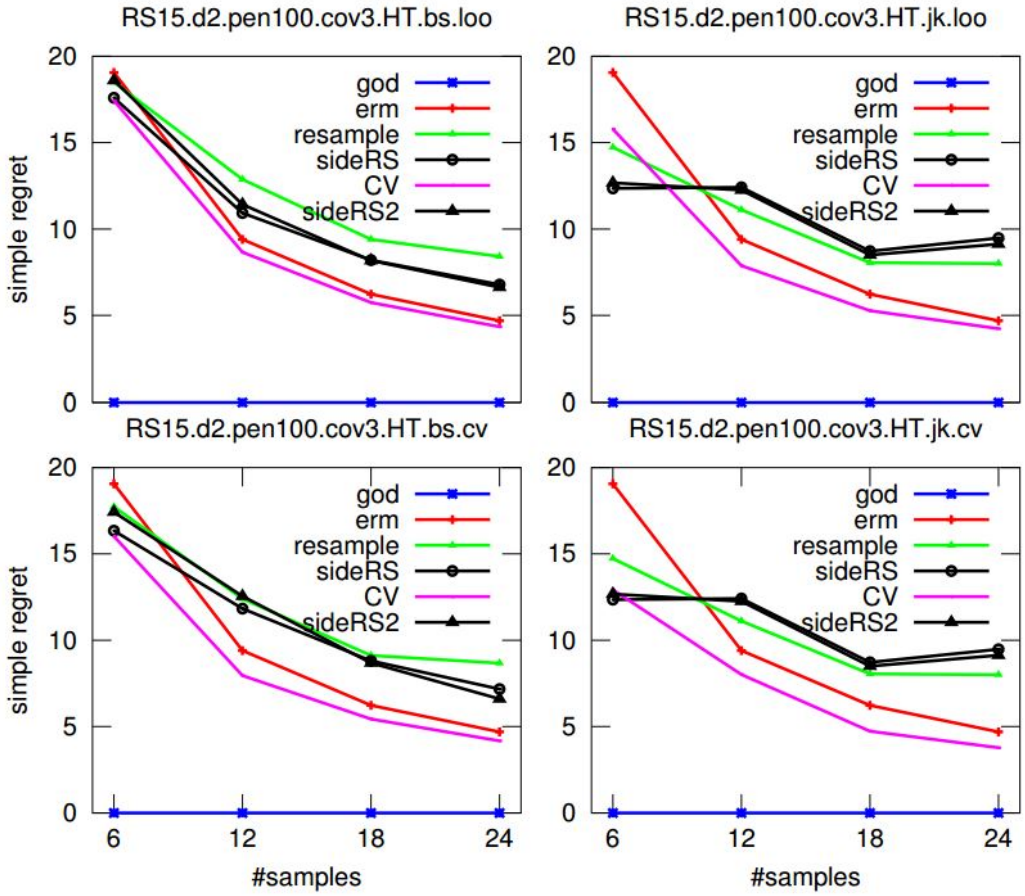


Figure 10.3: $d = 2$, $cov = 3$, $p = 100$, 15 resamplings for the bias reduction, heavy tail.

Name	bias correction	model selection	dim. reduction
$\hat{x}$	none	none	none
$\hat{x}_{BS}$	Bootstrap	none	none
$\hat{x}'_{BS}$	Bootstrap	none	absolute
$\hat{x}''_{BS}$	Bootstrap	none	relative
$\hat{x}_{JK}$	Jackknife	none	none
$\hat{x}'_{JK}$	Jackknife	none	absolute
$\hat{x}''_{JK}$	Jackknife	none	relative
$\hat{x}_{meta,L1}(\hat{x},\hat{x}_{JK},\hat{x}'_{JK})$	Jackknife	L1	absolute
$\hat{x}_{meta,L1}(\hat{x},\hat{x}_{BS},\hat{x}'_{BS})$	Bootstrap	L1	absolute
$\hat{x}_{meta,L2}(\hat{x},\hat{x}_{JK},\hat{x}'_{JK})$	Jackknife	L2	absolute
$\hat{x}_{meta,L2}(\hat{x},\hat{x}_{BS},\hat{x}'_{BS})$	Bootstrap	L2	absolute
$\hat{x}_{meta,L3}(\hat{x},\hat{x}_{JK},\hat{x}'_{JK})$	Jackknife	L3	absolute
$\hat{x}_{meta,L3}(\hat{x},\hat{x}_{BS},\hat{x}'_{BS})$	Bootstrap	L3	absolute

Table 10.1: Estimators compared in the experiments.

- L3 outperforms L1;
- Jackknife outperforms Bootstrap for bias-reduction;
- dimension reduction works well, in particular the absolute variant (Eq. 10.1).

Then Fig. 10.2 presents a case with a small penalty; the situation is far less satisfactory, though leave-three-out successfully often selects the naive ERM estimator. Compared to Fig. 10.1, the objective function is less skewed, hence the bias is much smaller and ERM performs well.

In Fig. 10.3, we see that model selection outperforms each of the models: this shows that the best estimate depends on the drawn sample, and that leave-three-out was able to "grasp" this effect.

A detailed analysis shows that Jackknife performs better than Bootstrap for bias reduction when bias correction was already not that bad; on the other hand, it makes results much worse in some cases in which they were already poor. This is somehow consistent with the literature (see Section 10.3.1).

L3 performs better than L1 (see Section 10.4.1).

The dimension reduction performs well in many cases. $\hat{x}'_{rs}$ outperforms the simple bias reduction $\hat{x}_{rs}$. So far, $\hat{x}''_{rs}$ is less efficient. This works even with *cov* small, i.e. inhomogeneous areas, as shown by Fig. 10.1. A small penalty $p = 3$ (Fig. 10.2) makes it hard for any algorithm to outperform the simple $\hat{x}$ estimate.

CV & margin method. With the simple Cross-Validation, results are usually positive, but robustness is a main issue. The new method (i.e. bias reduction estimates) should never be significantly worse than the old one (i.e. ERM). Typically, we want to reduce the risk of something going wrong as in Fig. 10.2, where the best result is indeed the simple ERM. The Penalized-Cross-Validation (Section 10.4.2) seems to be a good candidate to perform a reliable selection among the estimates. Furthermore, to ensure that the bias reduction method is always better or equal to ERM, we propose a third ingredient in the selection method, so that it has a bias in favour of ERM in case of doubt - this is the "margin" methodology (Section 10.4.4). This will automatically disable the bias correction for problems which are less risk sensitive - i.e. for which the bias of ERM is lower.

Hence, three model selection methods are compared in the setting of Section 10.5.1 to choose among $\hat{x}$ (default ERM method), $\hat{x}_{JK}$ (ERM estimate corrected by Jackknife), and $\hat{x}'_{JK}$ (ERM estimator corrected by Jackknife with dimension reduction as described in Section 10.3.2). These three selection method are (i) the classical CV (Section 10.4.1); (ii) the Penalized-Cross-Validation described in Section 10.4.2 and (iii) the Penalized-CV with margin described in Section 10.4.4.

There are 24 frameworks, combining dimension $d \in \{2,3,5,10\}$; $cov = d+1$, $cov = 10(d+1)$, $cov = 3000(d+1)$; 15 or 150 resamplings for the bias reduction, where d is the number of capacities to be estimated. Each dot in Figures 10.4 and 10.5 corresponds to one of the 24 corresponding frameworks, with results averaged over the different sample sizes, namely 6, 12, 18 and 24 samples. Each figure corresponds to 1 (top), 4 (middle) or 16 (bottom) splits in the Cross-Validation associated to the "meta" part (model selection); and we distinguish $L1$ (left), $L2$ (middle) or $L3$ (right). Figure 10.4 displays results of the CV method versus the Penalized-CV method. We use the default $C = 5/4 \times ((\#samples/k) - 1)$. Here $k \in \{1,2,3\}$ corresponds to the Lk considered. C is used as in Equation 10.3, overpenalization constant proposed in [Arlot, 2008].

In Fig.10.4, we see that difficult cases (Y-axis above 1) are $\circ$ and $+$, namely penalty 3 and 10 respectively: the classical method sometimes more than doubles

the *SR* (X-axis is limited at 2). With the Penalized-CV, the worst cases are around 1.5. The fact that problems occur around these values is reasonable: they are the cases in which ERM and Jackknife have comparable performance, so that CV might make a bad choice. In Fig.10.5, we see that the obtained CV method (Y-axis) is almost always < 1 (hence beneficial), though there are still a few cases with *SR* more than in the ERM case. The margin is applied in both cases (CV, and Penalized-CV.)

The Penalized-Cross-Validation outperforms the standard Cross-Validation - but there are still cases in which the simple ERM is the best, in particular for intermediate values of the penalty, when it is difficult to know the best among ERM and the bias-corrected variants.

It ensues that the best method is the Penalized-CV with 16 splits, $L3$, Jackknife, with 10% margin. Additional results are displayed in Table 10.2. These results indicate that ERM is the best for intermediate penalties. This fact is not surprising, these are the cases in which it is hard for CV methods to guess which estimator is the best. ERM is vastly outperformed in other cases (Fig. 10.5 and numbers in Table 10.2).

Table 10.2: Performance of Penalized-CV on various values of the penalty p in Eq. 10.4. The average normalized Simple Regret refers to the expected Simple Regret obtained by Penalized-CV divided by the expected Simple Regret of the ERM. Results are averaged over all 24 experiments for each value of p (all possibilities with $d \in \{2,3,5,10\}$, $cov \in \{(d+1), 10(d+1), 3000(d+1)\}$, 15 or 150 resamplings for the bias reduction). The standard deviations are at most 0.015.

penalty	0.1	1	3	10	30	100	3000
average normalized Simple Regret	.90	.98	1.03	1.04	.88	.66	.63

10.6 Conclusion

This chapter is devoted to the bias correction in empirical risk minimizers, including the multivariate case. Many studies are dedicated to capacity expansion planning for power systems [Saisirirat et al., 2013, Chaudry et al., 2014, NERC, 2013,

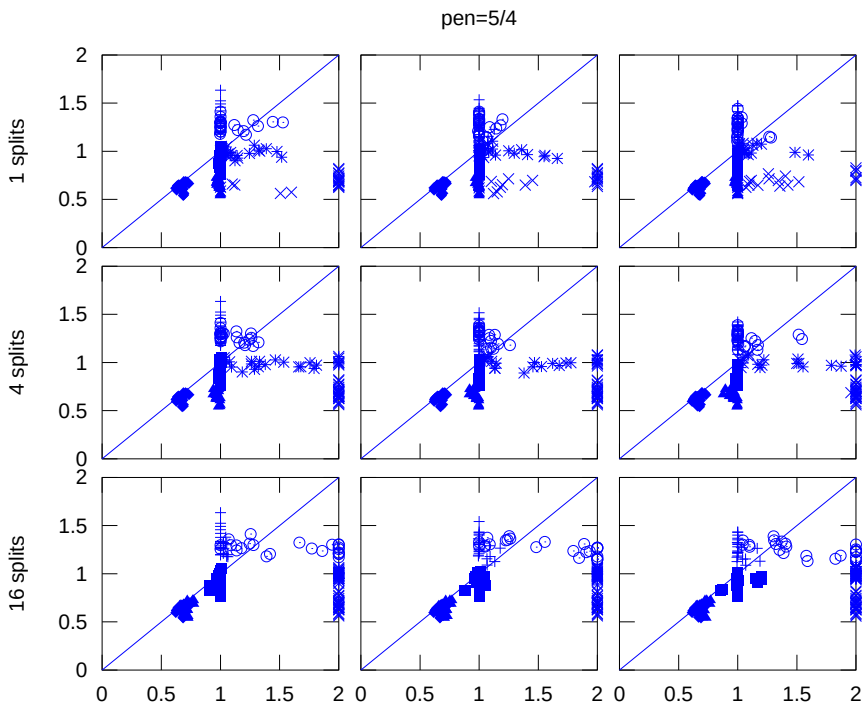


Figure 10.4: Experiments on Penalized-Cross-Validation. X-axis: *SR* obtained by the CV method divided by the *SR* of the naive ERM. Y-axis: *SR* obtained by the Penalized-CV method divided by the *SR* of the naive ERM. Each dot corresponds to one test case, averaged over the different sample sizes (6, 12, 18, 24). The markers $\times$, $*$, $\circ$, $+$, $\square$, $\hat{}$, $\Diamond$, stand for penalties $p = 0.1, 1, 3, 10, 30, 100, 3000$ respectively.

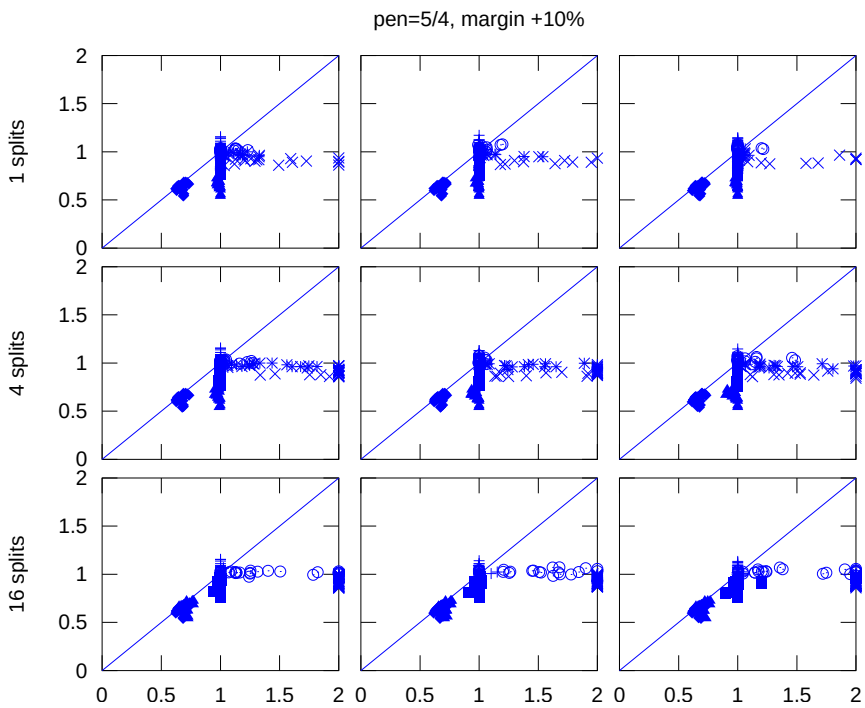


Figure 10.5: Similar to Figure 10.4, but the CV gives a 10% bonus (i.e. "margin" method) to ERM (i.e. $\gamma = 0.1$). Each dot corresponds to one test case, averaged over the different sample sizes (6, 12, 18, 24). The markers $\times$, $*$, $\circ$, $+$, $\square$, $\hat{}$, $\Diamond$, stand for penalties $p = 0.1, 1, 3, 10, 30, 100, 3000$ respectively.

Whitmarsh et al., 2012, Vassena et al., 2003, SYSTINT Workgroup, 2007]. To the best of our knowledge, bias correction has not been considered. Bias might be an overlooked serious issue in capacity estimation studies, which are a crucial part of power system optimization. We have investigated resamplings methods to reduce this bias.

In our experiments, Jackknife performed better than Bootstrap for correcting the bias. We improved the results thanks to a dimension reduction methodology. In high-dimensional cases, with homogeneous capacities to be estimated, averaging the bias correction over multiple capacities leads to a more efficient capacity correction than estimating each univariate correction alone. This technique provides an improved bias correction, and does not change the computational cost. The first variant of dimension reduction, termed absolute, was usually better in our experiments (Eq. 10.1, compared to Eq. 10.2).

For selecting estimators, Penalized-Cross-Validation outperformed the simple Cross-Validation. Furthermore, we developed the margin method for ensuring that the model selection is almost always better than the Sample Average Approximation method. Admittedly, this can reduce the average performance of the system; but it leads to the property that the meta-estimate is, in a stable manner, better or at least equal to the traditional estimate. We believe that such tricks are important for the acceptance of non-trivial statistical corrections.

Overall, statistical methods such as resampling can greatly increase the performance of capacity estimates - both for bias correction and for model selection. But there is a huge computational overhead. 100 samples for bias correction and 100 samples for model selection lead to a factor 10 000 on the computational cost. This is fortunately highly parallel, but the cost is far from being negligible.

Further work. The bias correction methods we propose are adaptations, to optimization, of general principles. A mathematical analysis exists for these tools. On the other hand, the margin method and the dimension reduction methods are new. Dimension reduction methods need mathematical analysis; maybe there are better solutions than the two extreme cases (absolute, as in Eq. 10.1, and relative, as in Eq. 10.2), for instance by considering groups of related capacities. Mathematical analysis might help to understand the bias/variance compromise in multivariate bias reduction of M-estimators (Eq. 10.1). The constant C in Eq. 10.3 is suggested only in a specific setting [Arlot, 2008]; we did not try any optimization of this constant, so that our results are principled, but improvements might be possible.

Considering years, in our archive of data, as independent, is an approximation. This is a reasonable assumption for some parts of the world but not for others: studying the impact of this lack of independence is another important further work [Yu, 1994].

Additional experiments are part of the agenda, including high-dimensional cases with hundreds of capacities.

III

Summary, Discussion and Perspectives

Chapter 11

Summary, Discussion, Perspectives

This Ph.D. thesis deals with noisy optimization problems that can be found in the context of power systems. It is divided into two parts. The first part is devoted to the continuous black-box noisy optimization problem. We have discussed the optimal rates of convergence of comparison-and-value-based algorithms and we have proposed a way to choose between different noisy optimization algorithms. The second part is dedicated to the treatment of 'delicate' black-box noise. The chapters in Part II are independent from each other and each of them handles a different aspect of optimization with respect to uncertainties. We review our results and propose some extensions. The end of the previous chapters already contains some conclusion and further work discussion. Thus, we discuss here broader topics for future research.

11.1 Part I: Summary and Discussion

11.1.1 Summary

Part I is dedicated to the handling of the expectation operator in Eq. 1.2, that we recall here:

$$i^* = \arg\min_{i \in \mathscr{I}} \max_{u \in \mathscr{U}} \min_{p \in \mathscr{P}_1} \min_{p' \in \mathscr{P}_2} \mathbb{E}_{\omega \sim \Pi} \, \textsc{Cost}(i, u, p, p', \omega). \tag{1.2}$$

The continuous black-box noisy problem takes an important place, in general, and in power system optimization, in particular, such as in Direct Policy Search

and Direct Model Predictive Control (see Chap. 8). We worked around the following research question:

Research Question #4

What are the optimal convergence rates of various families of black-box noisy continuous optimization algorithms?

The case of value-based algorithms in the framework of additive noise was already well-studied, as described in Chap. 2. We extended it to linear and multiplicative noise and detailed results regarding Cumulative Regret in Chap. 6. We incidentally recovered the known results in the additive case. In the linear and multiplicative cases, Simple and Cumulative Regrets can be optimal at the same time (with the same parametrization), contrarily to the additive case. In the linear case, $s(SR) = -1$ and $s(CR) = 0$ are optimal whereas in the multiplicative case, $s(SR) = -\infty$ and $s(CR) = 0$. In the latter case, we conjecture that the Simple Regret has a log-linear behaviour.

The case of comparison-based algorithms is more delicate: even in the noise-free setting, theoretical studies are tedious. We first showed in Chap. 3 that when dealing with small noise - smaller than multiplicative noise - it is easy to recover the same rate of convergence as in the noise-free case for an Evolution Strategy, provided that each search point is resampled.

We then determined a lower bound on the Simple Regret for a large class of comparison-based algorithms when confronted with additive noise, including the classical ESs. It was already known (see [Astete-Morales et al., 2014]), that ESs adapted to noise (resampled-ESs, with polynomial or exponential number of resamplings in function of the number of iterations) have a log-logarithmically convergence rate:

$$s(SR) = \frac{\log(SR_n)}{\log(n)} \to -\alpha, \quad \alpha > 0.$$

However, no information was available about the optimal value of α. In this framework, we have shown in Chap. 4 that this slope $-\alpha$ can not be better than $-1/2$ if the search points are sampled 'close' to the optimum, i.e. satisfying Eq. 4.5.

Was this lower bound universal, i.e. for every comparison-based algorithm? We found in Chap. 5 that, on the contrary, comparison-based algorithms can be fast: $-\alpha = -1$ for an optimization algorithm which samples far from the opti-

mum. We expect to find some ESs with such rate of convergence, as long as it samples search points far from the optimum.

Finally, we proposed a portfolio method to choose between different algorithms. This method guarantees the selection of the algorithm with the optimal rate of convergence for the noisy optimization problem at hand. It can also mitigate the possible 'bad luck' in the first iterations; this can be done by creating such a portfolio using several times the same algorithm, so that 'lucky runs' get rid of 'unlucky runs'. Importantly, detecting the best algorithm so far is very different from the noise-free case, see the LAG concept.

11.1.2 Perspectives & discussion

Adaptivity. As discussed previously, working on the adaptivity of an algorithm's parameters (e.g. step-size parameters, noise-related parameters) is an important task. We have suggested the use of a portfolio to select the best parametrization at hand. However, it is worth investigating other adaptation methods. We highlight the original work of [Massé and Ollivier, 2015] in the noise-free setting. They propose to adapt the step-size of a gradient descent by performing an online gradient descent on the step-size itself.

Surrogate model. We already mentioned that the use of surrogate models looks promising. In the noise-free case, [Auger et al., 2005] couples a quasi-random exploration and a local surrogate algorithm. It was shown that this memetic algorithm is super-linear, but not implementable in practice due to its prohibitive computational cost. We believe that taking the best of both worlds (evolutionary algorithms and value-based methods) could be relevant. However, we are not aware of any interesting results in the noisy setting.

Parallelization & quasi-random. Adapting efficient tools of the noise-free optimization framework is another possibility for improvement.

First of all, when an optimization algorithm calls the oracle several times within the same iteration step-size, it can be relevant to use parallelization. Therefore, p simultaneous calls to the objective function are possible when p is the number of processors or cores. This is particularly relevant if the computational cost is mainly in the objective function and is constant. In the noise-free case, [Fournier and Teytaud, 2010, Cauwet et al., 2015] have shown that Evolution Strategies, Differential Evolution and Particle Swarm Optimization have at

best a linear (in the number of processors) speed-up[1] when the order of magnitude of the population size is the same as the dimension and a logarithmic speed-up otherwise. Extending these results, especially implementing such a parallelization, to value-free and value-based noisy optimization algorithms could be a pertinent piece of work.

Regarding ESs in the noise-free setting, it is empirically efficient to replace random processes (in the mutation steps) by some quasi-random processes [Teytaud, 2008]. We believe that applying this technique to their noisy counterpart could bring some improvement as well.

Evaluation criteria. Relating to evaluation criteria (Simple Regret, Cumulative Regret, hitting point) of an algorithm, we want to point out some deep discrepancies between noisy and noise-free optimization. As it can be seen in Chap. 7 for instance, where one unique evaluation does not allow to choose the best of two noisy optimization algorithms (leading to the LAG concept) whereas comparisons in the noise-free setting are straightforward. Similar problems arise when we want to evaluate a noisy optimization algorithm within a testbed setting. How to evaluate when a precision is reached? In the noise-free framework, we can decide to stop the optimization once the algorithm reaches a given precision (e.g. optimal fitness within precision 10^{-8}). However in a noisy framework, we might have an excellent recommended point at a certain moment, and then move to a worse recommendation.

For example, in a noisy discrete setting, with domain $\mathscr{D} = \{0,1\}$ and a noisy objective function f defined by:

- $f(0) = \mathscr{G}$,
- $f(1) = \varepsilon + \mathscr{G}$,

where $\mathscr{G}$ is a standard Gaussian random variable. Consider the optimization algorithm which randomly chooses 0 with probability $1/2$ and 1 otherwise. Then the first hitting time is 1.5 on average, whereas for ε small, the time before 'knowing' which point among 0 and 1 is the best takes an arbitrarily large time.

Hence, it is necessary to find out good criteria in the context of noisy optimization. A discussion on this topic has been carried out in [Astete-Morales et al., 2016].

[1] The speed-up is the factor by which speed is improved when using p_2 processors instead of p_1.

Sampling far vs. sampling close to the optimum. In the noise-free case, the optimal strategy of a (1+1)-ES on the sphere function is to sample search points using a step-size proportional to the distance to the optimum. However, we conjecture that in the noisy additive case, an optimal rate of convergence can be reached if we sample 'far' from the optimum. In fact, in the noisy additive case, the closer we are to the optimum, the bigger the noise is. Sampling far from the optimum cancels this effect and gives more precise information. This conjecture is supported by the following observations:

- [Jamieson et al., 2012], when sampling at distance no more than $O(n^{-1/4})$, have a lower bound on the Simple Regret in $\Omega(n^{-1/2})$ on the family of strongly convex functions;
- Chap. 4 has shown that Evolution Strategies which sample 'close' to the optimum (proportional to the distance to the optimum) can not do better than a Simple Regret in $\Omega(n^{-1/2})$ on the family of quadratic functions;
- [Beyer, 1998] advises to 'mutate large but inherit small!' in a noisy setting;
- Chap. 5 provides a fast comparison-based algorithm - Simple Regret in $O(n^{-1})$ on quadratic functions - which samples 'far' from the optimum.

This feature is somehow the continuous optimization counterpart of the exploration vs. exploitation trade-off in multi-armed bandits.

Smart-grids and discrete optimization. A smart grid is composed of various electric appliances. Various challenging problems arise when dealing with it; among others, the need to maintain the same frequency for every component. There is a way to handle this problem with discrete optimization algorithms, such as $(1+1)$-EA, as follows. Each component is an agent, which can do an action among K possible actions. The objective function is the deviation to the target frequency. It depends on the actions of the n different components. In the asynchronized setting (the more realistic), at each iteration,

- play randomly uniformly among the K possible actions with a given probability;
- play the best action so far otherwise.

This is exactly the $(1+1)$-EA. We need to study the Cumulative Regret (which is the target in this setting) of such procedure[2]. What happens if the problem is

[2]Note that the Simple Regret is already provided by the literature.

modified, e.g. by adding new components into the grid or if the optimal reward changes? An insight could come from [Kötzing et al., 2015].

11.2 Part II: Summary and Discussion

11.2.1 Summary

Part II handles, independent of each other, the unit commitment problem in Chap. 8, non-stochastic uncertainties in Chap. 9 and investment problems with finite archive in Chap. 10.

Unit commitment problem

Chap. 8 focuses on the sub equation:

$$p^* = \arg\min_{p \in \mathscr{P}_1} \mathbb{E}_{\omega \sim \Pi} \textsc{Cost}(p, \omega).$$

The corresponding problem is recalled:

Research Question #1

How to handle the Unit Commitment Problem without assuming:

- the convexity or linearity of the cost function;
- the Markovianity (or equivalently, as discussed above, the moderate size complexity) of random process?

We investigated the Direct Model Predictive Control method to address this question. We proved that this method is able to recover the optimal policy without requiring linear or convex cost and transition functions and without requiring a Markovian, simple, random process. From an experimental point of view, this method is easy to implement on top of a classical MPC algorithm, anytime, stable and works in a polynomial time.

Non-stochastic uncertainties

Chap. 9 takes place in the setting of Eq 1.3:

$$i^* = \arg\min_{i \in \mathscr{I}} \max_{u \in \mathscr{U}} \textsc{Cost}(i, u), \tag{1.3}$$

and is related to the 2nd research question:

Research Question #2

How to approximate in a reasonable computational time a Nash equilibrium in the stochastic adversarial case?

We adapted in this chapter the Grigoriadis and Khachiyan's algorithm to the noisy setting with parallelization. The Nash criterion has the advantage of being less conservative than the Wald criterion, hence it is relevant in our power system optimization problem. However, as discussed in Section 1.2.3, a reflection has to be carried out to adapt such criterion in the power system setting.

Finite archive

Chap. 10 describes a method to tackle the problem of a finite, small archive when optimizing the capacities of a power system network. This problem was formulated in the following way:

Research Question #3

In the context of capacities expansion planning, how to reduce the bias resulting from a finite (small) archive?

We proposed to apply existing statistical methods, designed to reduce the bias, but they were never applied to such a problem, to the best of our knowledge. We then used a Penalized-Cross-Validation method which guarantees to always recommend an investment at least as good as the one that would be selected by the classical method used nowadays.

11.2.2 Perspectives & discussion

Model error vs. optimization error. In an optimization work, the error comes from two sides: the problem modelling and the optimization process. Whereas most works focus on reducing the optimization error, the problem modelling is often neglected and oversimplified models are used (convex costs, Markovian random processes). As developed in [Decock, 2014], it is crucial to optimize using a model as close as possible to the real model. This is the DMPC approach, covered

in Chapter 8. This approach should be generalized in optimization problems with similar features. A first step is to optimize using the simplified model but evaluate the cost of the resulted solution using the real model.

The importance of Cross-Validation. In power systems, the huge dimension and heavy constraints of the problem lead to a big computational cost, even with a smart optimization procedure. That is why, once the procedure returns an answer, the Cross-Validation part is often omitted. However, this phase should always be taken into account. If the complete method of Chap. 10 is too expensive for the problem at hand, we suggest to proceed as follows:

- increase the capacities returned by the optimization algorithm by 10%;
- test both the original capacities recommendation and the augmented by 10% one on some new scenarios/data;
- compare their respective cost.

In optimization against a finite number of scenarios, the capacities are often underestimated due to the bias - which is why we propose to increase them by 10%. This previous method allowed a Cross-Validation at a little cost.

Further experiments. This thesis is essentially theoretical. However, we believe that it is worth carrying out further experiments on the DMPC and the Cross-Validation method. Regarding DMPC, it would be interesting to compare it to SDDP in a setting where the cost and/or transition function are not linear/convex and/or the random process is non-Markovian. We would be interested in testing the validity of the Penalized-Cross-Validation method on real data. Note that it is relevant not only in optimizing a power system's capacities, but in other settings with similar characteristics. For instance, dimensioning capacities in a gas storage problem could benefit from this method as well.

11.3 Reflections on power systems optimization

In conclusion, we would like to emphasize a specificity of power systems that makes their optimization even more delicate: their confidential nature. It is indeed extremely difficult to obtain the required data (demands, weather, good modelling of a given power system network, etc ...) in order to test an optimization algorithm.

Power system interlocutors put forward two arguments: security and competitiveness. Regarding the first point, it is argued that, especially in the present context of e.g. terrorist threat, these sets of data in the wrong hands could lead to disasters. However, others actors of this sector acknowledge that someone highly determined to cause harm could find out enough information to do so anyway, so that the releasing of this data would not be that determinant. Regarding the second point, it takes place in a context of privatisation policy. An energy company releasing its data for free gives a competing company the opportunity to use it against them. That is, releasing data for free is incurring a loss. We advocate that the scope of power system challenges reaches beyond the commercial profit of a few companies, as the related pollution issues are costing society as a whole. We defend 'open source' and 'open data' systems, as we believe that if any researcher was able to easily test his ideas for the improvement of power system optimization, this field of research could jump forward several years.

Appendices

Appendix A

Summary of Notations

The main notations are summarized below. Note that some local notations, used only in one chapter, will be defined only locally, and not reported in the present Appendix. The main notations might as well be recalled through out the document.

$\mathbb{R}$	:=	set of real numbers
$\mathbb{R}_+$	:=	set of positive real numbers
$\mathbb{N}$	:=	$\{0,1,2,\dots\}$ set of natural numbers
$\mathbb{N}^*$	:=	$\{1,2,\dots\}$ set of positive natural numbers
f	:=	objective function
f'	:=	first derivative of function f, when f is unidimensional
$Df(x)$	:=	first derivative of function f at x (multidimensional case)
f''	:=	second derivative of function f, when f is unidimensional
d	:=	dimension of the search domain
$B_d(x,r)$	:=	ball of radius $r > 0$ centered at x in dimension d (the dimension might be omitted)
$\mathscr{D} \subset \mathbb{R}^d$	:=	search domain of the objective function
x^*	:=	optimum of the objective function
x_n	:=	search point used by the algorithm at iteration n or after n calls to the black-box, depending on the setting
y_n	:=	fitness value of x_n, possibly noisy

$\tilde{x}_n$	:=	recommendation of the optimum after n evaluations
ω	:=	random variable
$\mathscr{G}$	:=	Gaussian variable or vector (dimension, mean and standard deviation are sometimes specified)
$\|\cdot\|$	:=	Euclidean norm
$\lvert\cdot\rvert$	:=	Absolute value when applied to a real number Cardinality when applied to a set
$\langle\cdot,\cdot\rangle$	:=	Inner product
$\mathbb{P}$, (resp. $\mathbb{P}(\cdot\|\cdot)$)	:=	Probability (resp. conditional probability)
$\mathbb{E}$, (resp. $\mathbb{E}(\cdot\|\cdot)$)	:=	Expectation (resp. conditional expectation)
$\mathbb{E}_\omega$	:=	Expectation on the random variable ω
Var	:=	Variance of a random variable
$Q_{1-\delta}$	:=	Quantile $1-\delta$ of a random variable
UR_i	:=	$\|x_i - x^*\|$ Uniform Rate
SR_i	:=	$(\mathbb{E}_\omega f(\tilde{x}_i,\omega) - \mathbb{E}_\omega f(x^*,\omega))$ Simple Regret
CR_i	:=	$\sum_{j\leq i}\left(\mathbb{E}_\omega f(x_j,\omega) - \mathbb{E}_\omega f(x^*,\omega)\right)$ Cumulative Regret
$s(*R)$	:=	$\limsup_i \frac{\log(*R_i)}{\log(i)}$, $*R$ stand for SR, CR or UR, slope of the corresponding regret
$O(\cdot)$, $o(\cdot)$, $\Omega(\cdot),\Theta(\cdot)$	:=	Landau notations
$(e_j)_{j=1}^d$	:=	canonical orthonormal basis
$\lceil\cdot\rceil$ (resp. $\lfloor\cdot\rfloor$)	:=	ceiling (resp. floor) function
$x^{(i)}$	:=	i^{th} coordinate of vector x
M^{-1}	:=	inverse of matrix M, when M is invertible
M^t (resp. x^t)	:=	transpose of matrix M (resp. of vector x)
$M_{i,j}$	:=	entry in the ith row and j^{th} column of matrix M
$k \bmod l$	:=	remainder of the Euclidian division of k by l

Appendix B

Lower bound in $\Omega(n^{-1})$ or $\Omega(n^{-1/2})$: a discussion

We recall briefly the setting. f is a continuous noisy function for which, given a search point, an oracle returns the fitness value:

$$f(x,\omega) = f(x) + \omega$$

where $\mathbb{E}(\omega) = 0$ and $\mathbb{E}(\omega^2) \leq \sigma^2$. x^* denotes the optimum of the objective function, n the number of evaluations performed by the algorithm and $\tilde{x}_n$ is the approximation of x^* given by the optimization algorithm after spending n evaluations. We consider the Simple Regret:

$$SR_n = \mathbb{E}(f(\tilde{x}_n,\omega) - f(x^*,\omega))$$

and its corresponding *slope*: $s(SR) = \limsup_n \log(SR_n)/\log(n)$. In a noisy black-box continuous framework, what is the optimal lower bound of the Simple Regret?

B.1 Polyak-Tsybakov vs. Shamir

We recall Theorem 2.3.4, which states that for objective functions smooth enough, we can get a Simple Regret $s(SR) \to -1$, using the Polyak and Tsybakov algorithm.

Theorem 2.3.4 ([Polyak and Tsybakov, 1990], Simple Regret of Polyak-Tsybakov's algorithm)**.** *Assume that the objective function has a unique optimum at $x^*(f)$ and satisfies:*

1. *f has continuous partial derivatives up to order s inclusive, which satisfy the Hölder condition of order $\alpha \in (0,1]$;*

2. *$\forall\, x \in \mathbb{R}^d$, $(Df(x), x - x^*) \geq A_1 \|x - x^*\|^2$;*

3. *$\forall\, x, x' \in \mathbb{R}^d$, $\|Df(x) - Df(x')\| \leq A_2 \|x - x'\|$,*

where A_1 and A_2 are finite positive constants, $A_2 > A_1$. Let $\mathscr{F}$ denote the family of functions with a unique optimum satisfying these 3 conditions. Let $\beta = s + \alpha \geq 2$, $a_n = \frac{a}{n}$ and $c_n = \frac{c}{n^{1/2\beta}}$, $c > 0$, $a > (\beta - 1)/2A_1\beta$. Assume that the noise is additive, i.e. $\mathbb{E}(f(x,\omega) = f(x))$, with $\mathbb{E}(\omega) = 0$ and $\mathbb{E}(\omega^2)$ bounded. Assume that $\tilde{x}_n$ is obtained by the **Kiefer-Wolfowitz scheme** *(Eq. 2.14) with Eq. 2.22, it follows:*

$$\sup_n \sup_{f \in \mathscr{F}} n^{(\beta-1)/\beta} \mathbb{E}\|\tilde{x}_n - x^*(f)\|^2 < \infty \tag{2.23}$$

In particular, when f is smooth enough, we get $s(SR) = -(\beta - 1)/\beta$.

The optimality of this algorithm is also shown in the same paper [Polyak and Tsybakov, 1990]. We recall it as well.

Theorem 2.3.6 (Polyak-Tsybakov's lower bound)**.** *Let $\tilde{x}_n$ be any Borel function of $x_1, \ldots, x_n, y_1, \ldots, y_n$ where:*

- *$x_1 = h_1(\zeta)$, with ζ a random variable of arbitrary probability $\mathbb{P}_\zeta$ and h_1 a measurable function;*

- *$\forall\, i \in \{2, \ldots, n\}$, $x_{i+1} = h_i(x_1, \ldots, x_i, y_1, \ldots, y_i, \zeta)$, with ζ a random variable of arbitrary probability $\mathbb{P}_\zeta$ and h_i a measurable function;*

- *$\forall\, i \in \{2, \ldots, n\}$, $y_i = f(x_i) + \omega_i$, with $\omega_1, \ldots, \omega_n$ identically distributed with the distribution function g such that*

$$\int \ln(dg(u)/dg(u+t))dg(u) \leq I_0 t^2 \quad |t| \leq t_0, \tag{2.28}$$

  for some $0 < t_0 \leq \infty$ and $0 < I_0 < \infty$.

Let $\mathscr{F}$ be the family of functions as in Theorem 2.3.4, i.e., $\forall\, f \in \mathscr{F}$, f satisfies:

- *f has a unique optimum at $x^*(f)$;*

- *f has continuous partial derivatives up to order s inclusive, which satisfy the Hölder condition of order $\alpha \in (0,1]$;*

- $\forall\, x \in \mathbb{R}^d$, $(Df(x), x - x^*) \geq A_1 \|x - x^*\|^2$;
- $\forall\, x, x' \in \mathbb{R}^d$, $\|Df(x) - Df(x')\| \leq A_2 \|x - x'\|$,

where A_1 and A_2 are finite positive constants, $A_2 > A_1$ and $\beta = s + \alpha \geq 2$. Then

$$\inf_n \inf_{\tilde{x}_n} \sup_{f \in \mathscr{F}} n^{(\beta-1)/\beta} \mathbb{E}(\|\tilde{x}_n - x^*(f)\|^2) > 0. \tag{2.29}$$

However, [Shamir, 2013, Theorem 7] states that:

Theorem B.1.1 ([Shamir, 2013], Theorem 7)**.** *Let the number of rounds n be fixed. Then for any (possibly randomized) querying strategy, there exists a function f over $\mathbb{R}^d$ which is 0.5-strongly convex and 3.5-smooth; is 4-Lipschitz over the unit Euclidean ball; has a global minimum in the unit ball; and such that the resulting $\tilde{x}_n$ satisfies*

$$\mathbb{E}[f(\tilde{x}_n) - f(x^*)] \geq 0.0004 \min\left\{1, \sqrt{\frac{d^2}{n}}\right\}. \tag{B.1}$$

This suggests that $s(SR) \geq -1/2$, which seems at first sight in contradiction with Theorem 2.3.4. Does it come from a difference in the family of functions under consideration in the two settings? We will see that it, in fact, comes from another divergence of framework, that is called *asymptotic vs. non asymptotic setting*.

B.2 Asymptotic vs. non asymptotic setting

Let us investigate the family of functions in Theorem B.1.1. The proof in [Shamir, 2013] is carried out on the function:

$$f_e(x) = \|x\|^2 - \sum_{i=1}^{d} \frac{e_i x_i}{1 + \left(\frac{x_i}{e_i}\right)^2},$$

where e is uniformly distributed on $\{-\mu, +\mu\}^d$, μ being *a 'small' number, specified in the proof.*

Property B.2.1. *Function f_e satisfies conditions of Theorem 2.3.4 with $\beta > 2$. Hence, when applying Polyak and Tsybakov's Algorithm, we get $s(SR) < -1/2$ for this function.*

Proof. As it is stated in [Shamir, 2013], $f_e(x) = \sum_{i=1}^{d} g_{e_i}(x_i)$, where

$$g_a(x) = x^2 - \frac{ax}{1+(\frac{x}{a})^2}.$$

Hence, w.l.o.g, let us consider f_e in dimension 1: $f_e(x) = x^2 - \frac{ex}{1+(\frac{x}{e})^2}$.
We need to show that:

1. f_e has a unique minimum at the point x^*;
2. f_e is differentiable to order 3 inclusive;
3. $f_e'(x)(x-x^*) \geq A_1|x-x^*|^2$;
4. $|f_e'(x) - f_e'(x')| \leq A_2|x-x'|, \forall x, x' \in \mathbb{R}^d$.

1. f_e is minimized at ce where c can be computed explicitly (compute the real root of the polynomial function given by $f_e'(x) = 0$).
2. f_e is infinitely differentiable, as a rational function which denominator is never zero. Thus f_e is differentiable to order 3 inclusive.
3. [Shamir, 2013, Lemma 10] states that f_e is 0.5 strongly convex. Hence:

$$\begin{aligned} f_e(x^*) &\geq f_e(x) + f_e'(x)(x^*-x) + 0.25|x-x^*|^2 \\ f_e'(x)(x-x^*) &\geq f_e(x) - f_e(x^*) + 0.25|x-x^*|^2 \\ f_e'(x)(x-x^*) &\geq 0.25|x-x^*|^2 \text{ as } x^* \text{ is the minimum of } f_e. \end{aligned}$$

   Hence the inequality holds with $A_1 = 0.25$.

4. [Shamir, 2013, Lemma 10] states that $\forall x, |f_e''(x)| \leq 3.5$, hence we get $|f_e'(x) - f_e'(x')| \leq A_2|x-x'|, \forall x, x' \in \mathbb{R}^d$ with $A_2 = 3.5$.

□

So f_e satisfies condition of Theorem 2.3.4 with $\beta = 3$ (and actually $\beta \to \infty$), hence for n big enough, optimizing f_e with the Polyak and Tsybakov's algorithm will provide a Simple Regret satisfying $\mathbb{E}[f_e(\tilde{x}_n) - f_e(x^*)] = O(n^{-2/3})$. Where does this contradiction come from? A careful check of the proof of Theorem B.1.1 show that $\mu = n^{-1/4}$, (remember that f_e depends on $e \in \{-\mu, +\mu\}^d$). That is, $f_e := f_{e(n)}$.

Let us denote by $\mathscr{S}_n$ the set of functions which satisfies: $\forall f \in \mathscr{S}_n$, for any (possibly randomized) querying strategy,

$$\mathbb{E}[f(\tilde{x}_n) - f(x^*)] \geq \sqrt{\frac{d^2}{n}}. \tag{B.2}$$

Theorem B.1.1 states that for any n, $\mathscr{S}_n$ contains at least one strongly convex and smooth function. However, it does not contradict the fact that, for any strongly convex and smooth function f, $\exists n_o \; s.t. \forall n \geq n_0, f \notin \mathscr{S}_n$.

That is, both theorems hold, but differ from their settings. Polyak-Tsybakov's lower bound holds in an *asymptotic* setting whereas Shamir's lower bound holds for a fix number of evaluations n - i.e. a *non asymptotic* setting.

B.3 Illustration

In practice, we expect to observe, asymptotically, a slope -1 on a $\log - \log$ graph. We used Polyak-Tsybakov's algorithm to optimize the function:

$$f_{e(T)}(x) = \|x\|^2 - \sum_{i=1}^{d} \frac{e_i x_i}{1 + \left(\frac{x_i}{e_i}\right)^2},$$

where e is uniformly distributed on $\{-\mu, +\mu\}^d$, $\mu = T^{-1/4}$ for

$T \in \{10, 50, 100, 200, 300, 400, 500, 600, 10^3, 5 \times 10^3, 10^4, 5 \times 10^4, 10^5, 2 \times 10^5, 3 \times 10^5, 5 \times 10^5\}$.

However, it appeared that this parameter has no impact on the converge rate (though in theory, for a number of evaluations n smaller than T, the rate should not be better than $-1/2$), hence we display in Fig. B.1 the Simple Regret of $f_{e(T)}$ (in function of the number of evaluations) only for $T = 5000$. Function $f_e(T)$ should be similar to a sphere function $x \mapsto \|x\|^2$ for T big enough. We also display the Simple Regret of the sphere. We observe a slope -1 for both functions. Graphs are similar for other values of T.

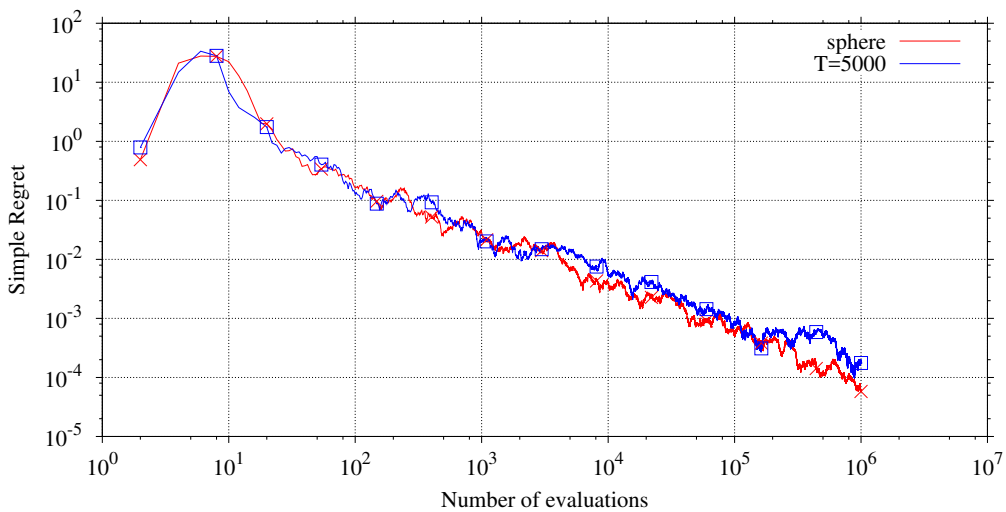


Figure B.1: Results of Polyak-Tsybakov's algorithm. Results are averaged over 11 runs in dimension $d = 2$. Parameters (see Theorem. 2.3.4) are: $a = 1$, $c = 1$, $1/2\beta = 0.1$ and the kernel K is chosen as described in [Polyak and Tsybakov, 1990] with $l = 3$. We observe a Simple Regret $s(SR) = -1$ for a number of evaluations big enough. Function $f_{e(T)}$ globally behaves as the sphere function.

Appendix C

Proofs of Sections 6.1.2 and 6.2.2 in Chapter 6

The following Lemmas are used to prove property 6.1.2. In this Section, consider functions SAMPLER and OPT as is Example 2, and an objective function f described by Eqs. 6.18 and 6.19. h denotes the Hessian of $\mathbb{E}f$ at 0, i.e. $h = (2c_{j,k})_{1\leq j,k\leq d}$. All results hold for C sufficiently small, and x verifying Eq. 6.20.

Lemma C.0.1 (Approximation lemma)**.** *With the definitions of Example 2, for any $j \in \{1,\dots,d\}$, the approximate gradient verifies:*

$$\hat{g}^{(j)} = \begin{cases} 2\sum\limits_{1\leq k\leq d} c_{j,k}x^{(k)} + \frac{\omega_1}{\sqrt{r}\sigma} & \text{if } \mathbb{E}_\omega f(x,\omega) = \sum\limits_{1\leq j,k\leq d} c_{j,k}x^{(j)}x^{(k)} \\ 2\sum\limits_{1\leq k\leq d} c_{j,k}x^{(k)} + O(\sigma^2) + \frac{\omega_1}{\sqrt{r}\sigma} & \text{otherwise} \end{cases} \tag{C.1}$$

where ω_1 is an independent noise, with $\mathbb{E}(\omega_1) = 0$ and $Var(\omega_1) = O(\sigma^{4z})$, $z \in \{0, 0.5, 1\}$.

For any $(j,k) \in \{1,\dots,d\}^2$, the approximate Hessian verifies:

$$\hat{h}_{j,k} = \begin{cases} 2c_{j,k} + \frac{\omega_2}{\sqrt{r}\sigma^2} & \text{if } \mathbb{E}_\omega f(x,\omega) = \sum\limits_{1\leq j,k\leq d} c_{j,k}x^{(j)}x^{(k)} \\ 2c_{j,k} + O(\sigma) + \frac{\omega_2}{\sqrt{r}\sigma^2} & \text{otherwise} \end{cases} \tag{C.2}$$

where ω_2 is an independent noise, independent of ω_1, with $\mathbb{E}(\omega_2) = 0$ and $Var(\omega_2) = O(\sigma^{4z})$, $z \in \{0, 0.5, 1\}$.

Proof. We provide a proof for the natural gradient $\hat{g}^{(j)}$. The proof is similar for natural Hessian. There are $\lfloor r/(2d^2) \rfloor$ revaluations per point. $\langle .,. \rangle$ is the inner product. By definition of the noisy objective function f in Property 6.1.2,

$$f(x,\omega) = \mathbb{E}_\omega f(x,\omega) + \omega,$$

with $\mathbb{E}_\omega f(x,\omega)$ as in Eq. 6.18 and ω a random variable s.t. $\mathbb{E}\omega = 0$ and $Var(\omega) = O(\|x\|^{4z})$, $z \in \{0, 0.5, 1\}$. In the same way,

$$\hat{g}^{(j)} = \mathbb{E}\hat{g}^{(j)} + \omega'$$

where ω' is a random variable s.t. $\mathbb{E}\omega' = 0$ and variance to calculate. By definition of the natural gradient in Ex. 2,

$$\begin{aligned}
\mathbb{E}\hat{g}^{(j)} &= \mathbb{E}\frac{\hat{y}_{j+} - \hat{y}_{j-}}{2\sigma} \\
&= \frac{1}{2\sigma}\left(\mathbb{E}_\omega f(x+\sigma e_j,\omega) - \mathbb{E}_\omega f(x-\sigma e_j,\omega)\right), \\
\mathbb{E}_\omega f(x+\sigma e_j,\omega) &= \sum_{1\le i,k\le d} c_{i,k} x^{(i)} x^{(k)} + 2\sigma \sum_{1\le k\le d} c_{j,k} x^{(k)} + c_{j,j}\sigma^2 \\
&+ \sum_{1\le i,k,l\le d} b_{i,k,l} x^{(i)} x^{(k)} x^{(l)} + \sigma \sum_{1\le k,l\le d} (b_{j,k,l} + b_{k,j,l} + b_{k,l,j}) x^{(k)} x^{(l)} \\
&+ \sigma^2 \sum_{1\le k\le d} (b_{j,j,k} + b_{k,j,j} + b_{j,k,j}) x^{(k)} + \sigma^3 b_{j,j,j} + o(\|x+\sigma e_j\|^3)
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}_\omega f(x+\sigma e_j,\omega) - \mathbb{E}_\omega f(x-\sigma e_j,\omega) = \\
4\sigma \sum_{1\le k\le d} c_{j,k} x^{(k)} + 2\sigma \underbrace{\sum_{1\le k,l\le d} (b_{j,k,l} + b_{k,j,l} + b_{k,l,j}) x^{(k)} x^{(l)}}_{=O(\|x\|^2)} \\
+ 2\sigma^3 b_{j,j,j} + o(\|x+\sigma e_j\|^3) - o(\|x-\sigma e_j\|^3).
\end{aligned}$$

By Eq. 6.20, $O(\|x\|^2) = O(\sigma^2)$ and $o(\|x \pm \sigma e_j\|^3) = O(\sigma^3)$. Then,

$$\mathbb{E}\hat{g}^{(j)} = \begin{cases} 2 \sum_{1\le k\le d} c_{j,k} x^{(k)} & \text{if } \mathbb{E} f(x,\omega) = \sum_{1\le j,k\le d} c_{j,k} x^{(j)} x^{(k)}, \\ 2 \sum_{1\le k\le d} c_{j,k} x^{(k)} + O(\sigma^2) & \text{otherwise.} \end{cases}$$

Now compute $Var(\omega') = Var(\hat{g}^{(j)})$.

$$
\begin{aligned}
Var(\hat{g}^{(j)}) &= Var\left(\frac{\hat{y}_{j+} - \hat{y}_{j-}}{2\sigma}\right) \\
&= \frac{Var\left(f(x+\sigma e_j, \omega) - f(x-\sigma e_j, \omega)\right)}{4\sigma^2 \lfloor r/(2d^2) \rfloor} \\
&= \frac{Var\left(f(x+\sigma e_j, \omega)\right) + Var\left(f(x-\sigma e_j, \omega)\right)}{4\sigma^2 \lfloor r/(2d^2) \rfloor}
\end{aligned}
$$

$Var\left(f(x+\sigma e_j, \omega)\right) \leq R\|x+\sigma e_j\|^{4z}$, $z \in \{0, 0.5, 1\}$ for a given $R > 0$ by Eq. 6.19.

$$
Var\left(f(x+\sigma e_j, \omega)\right) \quad \leq \quad \begin{cases} R & \text{if } z = 0, \\ R(\|x\|^2 + 2\langle x, \sigma e_j\rangle + \sigma^2) & \text{if } z = 0.5, \\ R(\|x\|^4 + 4(\sigma^2 + \|x\|^2)\langle x, \sigma e_j\rangle + \\ \quad 4\langle x, \sigma e_j\rangle^2 + 2\sigma^2\|x\|^2 + \sigma^4) & \text{if } z = 1. \end{cases}
$$

Hence using Eq. 6.20 and Cauchy-Schwarz inequality: $|\langle x, \sigma e_j\rangle| \leq \|x\|\sigma$, $Var\left(f(x+\sigma e_j, \omega)\right) = O(\sigma^{4z})$ and $Var\left(f(x-\sigma e_j, \omega)\right) = O(\sigma^{4z})$. Then $Var(\omega') = O(\frac{\sigma^{4z}}{r\sigma^2})$ and the expected result is obtained by putting $\omega' = \frac{\omega_1}{\sqrt{r}\sigma}$ with $Var(\omega_1) = O(\sigma^{4z})$. □

Lemma C.0.2. *There is $\sigma_0 > 0$ such that for all $\sigma < \sigma_0$, $\hat{h}$ is positive definite with its least eigenvalue greater than c_0 with probability at least $1 - O\left(\frac{\sigma^{4z-4}}{r}\right)$, where c_0 is as in the assumptions in Property 6.1.2.*

Proof. By Lemma C.0.1 (Eq. C.2), for any $(j,k) \in \{1, \ldots, d\}^2$, $\mathbb{E}\,\hat{h}_{j,k} = h_{j,k} + O(\sigma)$, where $\hat{h}$ is the approximation of the Hessian h, defined in Example 2 and σ is the step-size. So there exists a function $\sigma \mapsto f_h(\sigma)$ and a constant $R > 0$ such that $\mathbb{E}\,\hat{h}_{j,k} = h_{j,k} - f_h(\sigma)$ with $f_h(\sigma) \leq R|\sigma|$. Let c_0 be as in the assumptions in Property 6.1.2 and d the dimension. Then,

$$
\begin{aligned}
\mathbb{P}(|\hat{h}_{j,k} - h_{j,k}| \geq c_0/d) &= \mathbb{P}(|\hat{h}_{j,k} - h_{j,k} + f_h(\sigma) - f_h(\sigma)| \geq c_0/d) \\
&\leq \mathbb{P}(|\hat{h}_{j,k} - h_{j,k} + f_h(\sigma)| + f_h(\sigma) \geq c_0/d) \\
&= \mathbb{P}(|\hat{h}_{j,k} - \mathbb{E}\hat{h}_{j,k}| \geq c_0/d - f_h(\sigma)) \\
&\leq \frac{Var(\hat{h}_{j,k})}{(c_0/d - f_h(\sigma))^2} \text{ by applying Chebyshev's inequality}
\end{aligned}
$$

Also by Lemma C.0.1 (eq. C.2), $Var(\hat{h}_{j,k}) = Var(\frac{\omega_2}{\sqrt{r}\sigma^2}) = O(\frac{\sigma^{4z}}{r\sigma^4})$, where r is the number of revaluations. Furthermore, $(c_0/d - f_h(\sigma))^2 \geq (c_0/d - R|\sigma|)^2 > 0$ for all $\sigma < \frac{c_0}{dR} := \sigma_0$. So, $\forall (j,k) \in \{1,\dots,d\}^2$ we know that $\mathbb{P}(|\hat{h}_{j,k} - h_{j,k}| \geq c_0/d) = O(\frac{\sigma^{4z-4}}{r})$. Or equivalently

$$\mathbb{P}(|\hat{h}_{j,k} - h_{j,k}| \leq c_0/d) = 1 - O\left(\frac{\sigma^{4z-4}}{r}\right)$$

Since $\hat{h} - h$ is a symmetric matrix, then we deduce from Theorem 1 case (ii) in [Zhan, 2005] that $\lambda_d(\hat{h} - h) \geq -c_0$ with probability $1 - O\left(\frac{\sigma^{4z-4}}{r}\right)$, where we denote for any $d \times d$ matrix M its eigenvalues in decreasing order by $\lambda_1(M) \geq \cdots \geq \lambda_d(M)$. Using this and the fact that we assumed in Property 6.1.2 $\lambda_d(h) \geq 2c_0$, we have $\forall x \in \mathbb{R}^d$, $x \neq 0$,

$$\langle \hat{h}x, x\rangle = \langle (\hat{h} - h)x, x\rangle + \langle hx, x\rangle \geq -c_0\|x\|^2 + 2c_0\|x\|^2$$

$$\Leftrightarrow \frac{\langle \hat{h}x, x\rangle}{\langle x, x\rangle} \geq c_0$$

Since $\hat{h}$ is a Hermitian matrix, by the $\min - \max$ Theorem we know that $\lambda_d(\hat{h}) = \min\left\{\frac{\langle \hat{h}x,x\rangle}{\langle x,x\rangle}, x \neq 0\right\}$. Hence for all $\sigma > \sigma_0$, we have that $\lambda_d(\hat{h}) \geq c_0$ with probability $1 - O\left(\frac{\sigma^{4z-4}}{r}\right)$. □

Lemma C.0.3. (Good approximation of the optimum with the second order method) *Consider the context of Property 6.1.2. Then there exists a constant $K > 0$ such that for any pair of step size and number of revaluation (σ, r) that satisfies $\sigma^{6-4z} \leq K/r$, we have $\mathbb{E}(\|x^{\text{opt}}\|^2 | \mathscr{E}_{\hat{h}}^{c_0}) = O\left(\frac{\sigma^{4z-2}}{r}\right)$.*

Proof.

$$\begin{aligned}
\mathbb{E}(\|x^{\text{opt}}\|^2 | \mathscr{E}_{\hat{h}}^{c_0}) &= \mathbb{E}(\|\hat{h}^{-1}(\hat{h}x - \hat{g})\|^2 | \mathscr{E}_{\hat{h}}^{c_0}) \text{ by definition of } x^{\text{opt}}, \\
&\leq (1/c_0)^2 \mathbb{E}(\|\hat{h}x - \hat{g}\|^2 | \mathscr{E}_{\hat{h}}^{c_0})
\end{aligned}$$

using that $\forall x \in \mathbb{R}^d$, $x \neq 0$, $\|Mx\|^2 \leq (\lambda_1(M))^2 \|x\|^2$ and $\lambda_1(M^{-1}) = \frac{1}{\lambda_d(M)}$, where M is a real symmetric matrix. Under $\mathscr{E}_{\hat{h}}^{c_0}$, using Eqs. C.1 and C.2,

$$\mathbb{E}(\|\hat{h}x - \hat{g}\|^2) = \mathbb{E}\left\{\sum_{1\leq j\leq d}\left(\sum_{1\leq k\leq d} \hat{h}_{j,k}x^{(k)} - g^{(j)}\right)^2\right\},$$

$$
\begin{aligned}
&= \mathbb{E}\left\{\sum_{1\le j\le d}\left(\frac{\omega_2}{\sqrt{r}\sigma^2}\underbrace{\sum_{1\le k\le d} x^{(k)}}_{=O(\|x\|)}+O(\sigma)\underbrace{\sum_{1\le k\le d} x^{(k)}}_{=O(\|x\|)}+O(\sigma^2)+\frac{\omega_1}{\sqrt{r}\sigma}\right)^2\right\} \\
&= d\mathbb{E}\left\{\left(O(\sigma^2)+\frac{\omega_1}{\sqrt{r}\sigma}+\frac{\omega_2}{\sqrt{r}\sigma}\right)^2\right\} \text{ using } \|x\|\le C\sigma, \\
&= O(\sigma^4)+O\left(\frac{\sigma^{4z}}{r\sigma^2}\right) \text{ using } \mathbb{E}(\omega_1)=\mathbb{E}(\omega_2)=0, \\
&\quad Var(\omega_1)=Var(\omega_2)=O(\sigma^{4z}) \text{ and independence}, \\
&= O\left(\frac{\sigma^{(4z-2)}}{r}\right) \text{ if } \sigma^{4-(4z-2)}\le K/r,
\end{aligned}
$$

which is the expected result. □

Remark C.0.1. *In Lemma C.0.3, if $\mathbb{E}f$ is simply quadratic, i.e. $\forall(j,k,l)\in\{1,\dots,d\}^3$, $b_{j,k,l}=0$, the assumption $\sigma^{6-4z}=O(1/r)$ is unnecessary.*

Appendix D

Appendix: Proof of Lemma 8.3.1

Lemma 8.3.1. *There exist $d+1$ unit vectors $w_1, \dots, w_{d+1}$ in $\mathbb{R}^d$, and there exists a constant $c > 0$ such that, for any unit vector $u \in \mathbb{R}^d$, there exist $i \in \{1, \dots, d+1\}$ such that $\langle u, w_i \rangle > c$.*

Proof. $x^{(k)}$ is the k-th coordinate of vector x.

Consider $\mathbb{S} = \{x \in \mathbb{R}^{d+1} \mid \sum_{i=1}^{d+1} (x^{(i)})^2 = d^2 + d \text{ and } \sum_{i=1}^{d+1} x^{(i)} = 0\}$. $\mathbb{S}$ is the intersection of a d-dimensional sphere of $\mathbb{R}^{d+1}$ and of a hyperplane in dimension $d+1$; it is therefore a $(d-1)$-dimensional sphere of a d-dimensional Euclidean space (which is the hyperplane $H = \{x; \sum_{i=1}^{d+1} x^{(i)} = 0\}$ of $\mathbb{R}^{d+1}$). We show the result in this d-dimensional Euclidean space.

Let us consider $w_1, \dots, w_{d+1}$ defined by:

$$\begin{aligned} \forall i \in \{1, \dots, d+1\}, \qquad & w_i \in \mathbb{R}^{d+1} \\ \text{and } \forall j \in \{1, \dots, d+1\}, \qquad & w_i^{(j)} = \begin{cases} d & \text{if } j = i, \\ -1 & \text{otherwise.} \end{cases} \end{aligned}$$

The w_i are elements of $\mathbb{S}$. They are a regular simplex. For any $x \in \mathbb{S}$, we define $v(x) = \max_{i \in \{1, \dots, d+1\}} \langle x, w_i \rangle$. Let us show that for any $x \in \mathbb{S}$, $v(x) > 0$.

Without loss of generality, let us assume that $x^{(1)} > x^{(j)}$, for any $j \in \{2, \dots, d+1\}$ (otherwise just permute coordinates, the problem is invariant by such permutations).

$$\langle x, w_1 \rangle = (d+1)x^{(1)} - \underbrace{\sum_{i=1}^{d+1} x^{(i)}}_{0, \text{ because } \mathbb{S} \subset H},$$

so for any $x \in \mathbb{S}$, $v(x) = (d+1) \max_{i \in \{1,\dots,d+1\}} x^{(i)}$. By definition of $\mathbb{S}$, $\max_{i \in \{1,\dots,d+1\}} x^{(i)} > 0$, therefore $v(x) > 0$. Since $\mathbb{S}$ is a compact Hausdorff space, v reaches its lower bound on $\mathbb{S}$: there exists $c > 0$ such that $\forall x \in \mathbb{S}$, $v(x) \geq c$.

We have the above conclusion for the hyperplane $\sum_{i=1}^{d+1} x^{(i)} = 0$ of $\mathbb{R}^{d+1}$, thus we have the same conclusion for the domain $\mathbb{R}^d$. We have concluded the proof for x such that $\|x\|^2 = d^2 + d$; a fortiori the result holds for x such that $\|x\| = 1$.

$\square$

Bibliography

[Aha, 1992] Aha, D. W. (1992). Generalizing from Case Studies: A Case Study. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 1–10. Morgan Kaufmann.

[Aizawa and Wah, 1993] Aizawa, A. N. and Wah, B. W. (1993). Dynamic control of genetic algorithms in a noisy environment. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, pages 48–55. Morgan Kaufman.

[Aizawa and Wah, 1994] Aizawa, A. N. and Wah, B. W. (1994). Scheduling of Genetic Algorithms in a Noisy Environment. *Evolutionary Computation*, 2(2):97–122.

[Akimoto et al., 2015] Akimoto, Y., Astete-Morales, S., and Teytaud, O. (2015). Analysis of runtime of optimization algorithms for noisy functions over discrete codomains. *Journal of Theoretical Computer Science (TCS)*, 605:42–50.

[Arlot, 2008] Arlot, S. (2008). V-fold cross-validation improved: V-fold penalization. Preprint.

[Arlot and Celisse, 2010] Arlot, S. and Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Survey*, 4:40–79.

[Armstrong et al., 2006] Armstrong, W., Christen, P., McCreath, E., and Rendell, A. P. (2006). Dynamic Algorithm Selection Using Reinforcement Learning. In *International Workshop on Integrating AI and Data Mining*, pages 18–25.

[Arnold, 2002] Arnold, D. V. (2002). *Noisy Optimization with Evolution Strategies*. Genetic Algorithms and Evolutionary Computation Series. Kluwer Academic Publishers.

[Arnold and Beyer, 2000a] Arnold, D. V. and Beyer, H. (2000a). Local Performance of the $(\mu/\mu_i, \mu)$-ES in a Noisy Environment. In *Proceedings of the Sixth Workshop on Foundations of Genetic Algorithms, Charlottesville, VA, USA, July 21-23, 2000*, pages 127–141.

[Arnold and Beyer, 2000b] Arnold, D. V. and Beyer, H.-G. (2000b). Efficiency and Mutation Strength Adaptation of the $(\mu/\mu_I, \lambda)$-ES in a Noisy Environment. In Schoenauer, M., Deb, K., Rudolph, G., Yao, X., Lutton, E., Merelo, J. J., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 18–20, 2000 Proceedings*, pages 39–48. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Arnold and Beyer, 2001] Arnold, D. V. and Beyer, H.-G. (2001). Investigation of the (μ, λ)-ES in the presence of noise. In *IEEE Congress on Evolutionary Computation 2001*, volume 1, pages 332–339.

[Arnold and Beyer, 2002] Arnold, D. V. and Beyer, H.-G. (2002). Local performance of the (1 + 1)-ES in a noisy environment. *IEEE Transactions on Evolutionary Computation*, 6(1):30–41.

[Arnold and Beyer, 2003] Arnold, D. V. and Beyer, H.-G. (2003). On the effects of outliers on evolutionary optimization. In Liu, J., Cheung, Y.-m., and Yin, H., editors, *Intelligent Data Engineering and Automated Learning: 4th International Conference, IDEAL 2003, Hong Kong, China, March 21-23, 2003. Revised Papers*, pages 151–160. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Arnold and Beyer, 2006] Arnold, D. V. and Beyer, H.-G. (2006). A general noise model and its effects on evolution strategy performance. *IEEE Transactions on Evolutionary Computation*, 10(4):380–391.

[Astete-Morales et al., 2015a] Astete-Morales, S., Cauwet, M.-L., Liu, J., and Teytaud, O. (2015a). Simple and Cumulative Regret for Continuous Noisy Optimization. *Journal of Theoretical Computer Science (TCS)*, 617:12–27.

[Astete-Morales et al., 2015b] Astete-Morales, S., Cauwet, M.-L., and Teytaud, O. (2015b). Evolution strategies with additive noise: A convergence rate lower bound. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, FOGA '15, pages 76–84, New York, NY, USA. ACM.

[Astete-Morales et al., 2016] Astete-Morales, S., Cauwet, M.-L., and Teytaud, O. (2016). Analysis of Different Types of Regret in Continuous Noisy Optimization. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016*, pages 205–212.

[Astete-Morales et al., 2014] Astete-Morales, S. C., Liu, J., and Teytaud, O. (2014). Log-log Convergence for Noisy Optimization. In Legrand, P., Corsini, M., Hao, J., Monmarché, N., Lutton, E., and Schoenauer, M., editors, *Artificial Evolution - 11th International Conference, Evolution Artificielle, EA 2013, Bordeaux, France, October 21-23, 2013. Revised Selected Papers*, volume 8752 of *Lecture Notes in Computer Science*, pages 16 –28, Bordeaux, France. Springer.

[Aubin, 2009] Aubin, J.-P. (2009). *Viability Theory*. Birkhäuser Boston, Boston.

[Auer, 2002] Auer, P. (2002). Using Confidence Bounds for Exploitation-Exploration Trade-offs. *Journal of Machine Learning Research*, 3:397–422.

[Auer et al., 2002] Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002). Finite time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2/3):235–256.

[Auer et al., 1995] Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (1995). Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of the 36th Annual Symposium on Foundations of Computer Science*, pages 322–331. IEEE Computer Society Press, Los Alamitos, CA.

[Auger, 2005] Auger, A. (2005). Convergence results for the $(1,\lambda)$-SA-ES using the theory of ϕ-irreducible Markov chains. *Theoretical Computer Science*, 334(1-3):35–69.

[Auger, 2016] Auger, A. (2016). *Analysis of Comparison-based Stochastic Continuous Black-Box Optimization Algorithms*. Habilitation à diriger des recherches en mathématiques et en informatique, Université Paris Sud.

[Auger et al., 2010a] Auger, A., Brockhoff, D., and Hansen, N. (2010a). Benchmarking the (1, 4)-CMA-ES with mirrored sampling and sequential selection on the noisy BBOB-2010 testbed. In *Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, Portland, Oregon, USA, July 7-11, 2010, Companion Material*, pages 1625–1632.

[Auger et al., 2010b] Auger, A., Brockhoff, D., and Hansen, N. (2010b). Investigating the impact of sequential selection in the (1, 4)-CMA-ES on the noisy BBOB-2010 testbed. In *Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, Portland, Oregon, USA, July 7-11, 2010, Companion Material*, pages 1611–1616.

[Auger et al., 2010c] Auger, A., Brockhoff, D., and Hansen, N. (2010c). Mirrored variants of the (1, 2)-CMA-ES compared on the noisy BBOB-2010 testbed. In *Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, Portland, Oregon, USA, July 7-11, 2010, Companion Material*, pages 1575–1582.

[Auger and Hansen, 2013] Auger, A. and Hansen, N. (2013). Linear Convergence on Positively Homogeneous Functions of a Comparison Based Step-Size Adaptive Randomized Search: the (1+1) ES with Generalized One-fifth Success Rule. *Preprint*. arXiv: 1310.8397.

[Auger et al., 2005] Auger, A., Schoenauer, M., and Teytaud, O. (2005). Local and global oder 3/2 convergence of a surrogate evolutionary algorithm. In *Genetic and Evolutionary Computation Conference, GECCO 2005, Proceedings, Washington DC, USA, June 25-29, 2005*, pages 857–864, Washington. ACM.

[Autorita per l'Energia Elettrica e il Gas, 2004] Autorita per l'Energia Elettrica e il Gas, C. (2004). Report on the events of september 28th, 2003 culminating in the separation of the italian power system from the other UCTE networks. Accessed: 2010-09-09.

[Bach and Perchet, 2016] Bach, F. and Perchet, V. (2016). Highly-Smooth Zeroth Order Online Optimization. In 29th *Annual Conference on Learning Theory (COLT)*, pages 257–283, New York, United States.

[Bellman, 1957] Bellman, R. (1957). *Dynamic Programming*. Princeton University Press.

[Belloni et al., 2003] Belloni, A., Lima, A. D. S., Maceira, M. P., and Sagastizábal, C. (2003). Bundle Relaxation and Primal Recovery in Unit Commitment Problems. The Brazilian Case. *Annals of Operations Research*, 120(1):21–44.

[Benders, 1962] Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252.

[Bertsekas, 1995] Bertsekas, D. P. (1995). *Dynamic Programming and Optimal Control*, volume I and II. Athena Scientific.

[Bertsekas, 2005] Bertsekas, D. P. (2005). Dynamic programming and suboptimal control: A survey from ADP to MPC. *European Journal of Control*, 11(4-5):310–334.

[Bertsimas et al., 2013] Bertsimas, D., Litvinov, E., Sun, X. A., Zhao, J., and Zheng, T. (2013). Adaptive robust optimization for the security constrained unit commitment problem. *IEEE Transactions on Power Systems*, 28(1):52–63.

[Bertsimas and Sim, 2004] Bertsimas, D. and Sim, M. (2004). The price of robustness. *Operations Research*, 52(1):35–53.

[Bertsimas and Tsitsiklis, 1997] Bertsimas, D. and Tsitsiklis, J. (1997). *Introduction to Linear Optimization*. Athena Scientific, 1st edition.

[Beyer, 1993] Beyer, H.-G. (1993). Toward a Theory of Evolution Strategies: Some Asymptotical Results from the $(1,+\lambda)$-theory. *Evolutionary Computation*, 1(2):165–188.

[Beyer, 1998] Beyer, H.-G. (1998). Mutate large, but inherit small! on the analysis of rescaled mutations in $(\tilde{1},\tilde{\lambda})$-ES with noisy fitness data. In Eiben, A. E., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature — PPSN V: 5th International Conference Amsterdam, The Netherlands September 27–30, 1998 Proceedings*, pages 109–118. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Beyer, 2001] Beyer, H.-G. (2001). *The Theory of Evolution Strategies*. Natural Computing Series. Springer-Verlag, Netherlands.

[Beyer, 2004] Beyer, H.-G. (2004). Actuator noise in recombinant evolution strategies on general quadratic fitness models. In Deb, K., editor, *Genetic and Evolutionary Computation – GECCO 2004: Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004. Proceedings, Part I*, pages 654–665. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Beyer and Schwefel, 2002] Beyer, H.-G. and Schwefel, H.-P. (2002). Evolution Strategies – A Comprehensive Introduction. *Natural Computing*, 1(1):3–52.

[Blum, 1954a] Blum, J. R. (1954a). Approximation Methods which Converge with Probability one. *The Annals of Mathematical Statistics*, 25(2):382–386.

[Blum, 1954b] Blum, J. R. (1954b). Multidimensional Stochastic Approximation Methods. *The Annals of Mathematical Statistics*, 25(4):737–744.

[Borrett and Tsang, 1996] Borrett, J. and Tsang, E. P. K. (1996). Towards a Formal Framework for Comparing Constraint Satisfaction Problem Formulations. Technical Report CSM-264, University of Essex, Department of Computer Science.

[Branke and Schmidt, 2003] Branke, J. and Schmidt, C. (2003). Selection in the presence of noise. In Cantú-Paz, E., Foster, J. A., Deb, K., Davis, L. D., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Standish, R., Kendall, G., Wilson, S., Harman, M., Wegener, J., Dasgupta, D., Potter, M. A., Schultz, A. C., Dowsland, K. A., Jonoska, N., and Miller, J., editors, *Genetic and Evolutionary Computation — GECCO 2003: Genetic and Evolutionary Computation Conference Chicago, IL, USA, July 12–16, 2003 Proceedings, Part I*, pages 766–777. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Branke and Schmidt, 2004] Branke, J. and Schmidt, C. (2004). Sequential sampling in noisy environments. In Yao, X., Burke, E. K., Lozano, J. A., Smith, J., Merelo-Guervós, J. J., Bullinaria, J. A., Rowe, J. E., Tiňo, P., Kabán, A., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature - PPSN VIII: 8th International Conference, Birmingham, UK, September 18-22, 2004. Proceedings*, pages 202–211. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Branke et al., 2001] Branke, J., Schmidt, C., and Schmeck, H. (2001). Efficient fitness estimation in noisy environments. In *Proceedings of Genetic and Evolutionary Computation*, pages 243–250.

[Bubeck et al., 2009] Bubeck, S., Munos, R., and Stoltz, G. (2009). Pure exploration in multi-armed bandits problems. In *Algorithmic Learning Theory, 20th International Conference, ALT 2009, Porto, Portugal, October 3-5, 2009. Proceedings*, pages 23–37.

[Bubeck et al., 2011] Bubeck, S., Munos, R., and Stoltz, G. (2011). Pure exploration in finitely-armed and continuous-armed bandits. *Theoretical Computer Science*, 412(19):1832–1852.

[Burkholder, 1956] Burkholder, D. L. (1956). On a Class of Stochastic Approximation Processes. *The Annals of Mathematical Statistics*, 27(4):1044–1059.

[Cantú-Paz, 2004] Cantú-Paz, E. (2004). Adaptive sampling for noisy problems. In *Genetic and Evolutionary Computation - GECCO 2004, Genetic and Evolutionary Computation Conference, Seattle, WA, USA, June 26-30, 2004, Proceedings, Part I*, pages 947–958.

[Cappé et al., 2013] Cappé, O., Garivier, A., Maillard, O.-A., Munos, R., and Stoltz, G. (2013). Kullback-Leibler Upper Confidence Bounds for Optimal Sequential Allocation. *The Annals of Statistics*, 41(3):1516–1541.

[Castronovo et al., 2016] Castronovo, M., Ernst, D., Couëtoux, A., and Fonteneau, R. (2016). Benchmarking for Bayesian Reinforcement Learning. *PLoS ONE*, 11(6):1–25.

[Cauwet, 2014] Cauwet, M. (2014). Noisy optimization: Convergence with a fixed number of resamplings. In *Applications of Evolutionary Computation - 17th European Conference, EvoApplications 2014, Granada, Spain, April 23-25, 2014, Revised Selected Papers*, pages 603–614.

[Cauwet et al., 2016a] Cauwet, M.-L., Christophe, J.-J., Decock, J., Liu, J., and Teytaud, O. (2016a). A Consistent Model Predictive Control. *To be submitted.*

[Cauwet et al., 2016b] Cauwet, M.-L., Liu, J., Rozière, B., and Teytaud, O. (2016b). Algorithm portfolios for noisy optimization. *Annals of Mathematics and Artificial Intelligence*, 76(1-2):143–172.

[Cauwet et al., 2014] Cauwet, M.-L., Liu, J., and Teytaud, O. (2014). Algorithm portfolios for noisy optimization: Compare solvers early. In Pardalos, M. P., Resende, G. M., Vogiatzis, C., and Walteros, L. J., editors, *Learning and Intelligent Optimization: 8th International Conference, Lion 8, Gainesville, FL, USA, February 16-21, 2014. Revised Selected Papers*, pages 1–15. Springer International Publishing, Cham.

[Cauwet et al., 2016c] Cauwet, M.-L., St-Pierre, D. L., and Teytaud, O. (2016c). Stochastic zero-sum games. *To be submitted.*

[Cauwet and Teytaud, 2016a] Cauwet, M.-L. and Teytaud, O. (2016a). Multivariate bias reduction in capacity expansion planning. In *Power Systems Computation Conference, PSCC 2016, Genoa, Italy, June 20-24, 2016*, pages 1–8.

[Cauwet and Teytaud, 2016b] Cauwet, M.-L. and Teytaud, O. (2016b). Noisy optimization: Fast convergence rates with comparison-based algorithms. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference, Denver, CO, USA, July 20 - 24, 2016*, pages 1101–1106.

[Cauwet et al., 2015] Cauwet, M.-L., Teytaud, O., Chiu, S.-Y., Lin, K.-M., Yen, S.-J., Saint-Pierre, D. L., and Teytaud, F. (2015). Parallel Evolutionary Algorithms Performing Pairwise Comparisons. In He, J., Jansen, T., Ochoa, G., and Zarges, C., editors, *Foundations of Genetic Algorithms*, Foundations of Genetic Algorithms, pages 99–113, Aberystwyth, United Kingdom. ACM.

[Chaloner, 1989] Chaloner, K. (1989). Bayesian design for estimating the turning point of a quadratic regression. *Communications in Statistics - Theory and Methods*, 18(4):1385–1400.

[Chapel and Deffuant, 2006] Chapel, L. and Deffuant, G. (2006). SVM viability controller active learning. In *Kernel machines and Reinforcement Learning Workshop, Pittsburgh, PA*.

[Chaslot et al., 2008] Chaslot, G. M. J.-b., Win, M. H. M., and Herik, H. J. V. D. (2008). Parameter tuning by the cross-entropy method. In *Proceedings of the 8th European Workshop on Reinforcement Learning*.

[Chatzivasileiadis et al., 2013] Chatzivasileiadis, S., Ernst, D., and Andersson, G. (2013). The global grid. *Renewable Energy*, 87:372–383.

[Chaudry et al., 2014] Chaudry, M., Jenkins, N., Qadrdan, M., and Wu, J. (2014). Combined gas and electricity network expansion planning. *Applied Energy*, 113(C):1171–1187.

[Chen, 1988] Chen, H. (1988). Lower rate of convergence for locating a maximum of a function. *The Annals of Statistics*, 16(3):1330–1334.

[Cheney and Goldstein, 1959] Cheney, E. W. and Goldstein, A. A. (1959). Newton's method for convex programming and Tchebycheff approximation. *Numerische Mathematik*, 1(1):253–268.

[Chotard, 2015] Chotard, A. (2015). *Markov chain Analysis of Evolution Strategies*. Thesis, Université Paris Sud - Paris XI.

[Chotard et al., 2012] Chotard, A. A., Auger, A., and Hansen, N. (2012). Cumulative Step-size Adaptation on Linear Functions: Technical Report. *Preprint*. arXiv: 1206.1208.

[Christophe et al., 2014] Christophe, J.-J., Decock, J., and Teytaud, O. (2014). Direct model predictive control. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, Bruges, Belgique.

[Chung, 1954] Chung, K. L. (1954). On a Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 25(3):463–483.

[Conca, 2012] Conca, J. (2012). How Deadly Is Your Kilowatt? We Rank The Killer Energy Sources. `http://www.forbes.com/sites/jamesconca/2012/06/10/energys-deathprint-a-price-always-paid`. Accessed: 2016-04-18.

[Corus et al., 2014] Corus, D., Dang, D.-C., Eremeev, A. V., and Lehre, P. K. (2014). Level-based analysis of genetic algorithms and other search processes. In Bartz-Beielstein, T., Branke, J., Filipico, B., and Smith, J., editors, *Parallel Problem Solving from Nature - PPSN XIII*, volume 8672 of *Lecture Notes in Computer Science*, pages 912–921. Springer International Publishing.

[Couëtoux, 2013] Couëtoux, A. (2013). *Monte Carlo Tree Search for Continuous and Stochastic Sequential Decision Making Problems*. Theses, Université Paris Sud - Paris XI.

[Coulom, 2011] Coulom, R. (2011). CLOP: Confident Local Optimization for Noisy Black-Box Parameter Tuning. In Herik, H. J. v. d. and Plaat, A., editors, *Advances in Computer Games*, number 7168 in Lecture Notes in Computer Science, pages 146–157. Springer Berlin Heidelberg. DOI: 10.1007/978-3-642-31866-5_13.

[Coulom et al., 2011] Coulom, R., Rolet, P., Sokolovska, N., and Teytaud, O. (2011). Handling Expensive Optimization with Large Noise. In *Proceedings of the 11th workshop on Foundations of genetic algorithms*, pages 61–68, Austria. ACM.

[Dang and Lehre, 2015] Dang, D. and Lehre, P. K. (2015). Efficient optimisation of noisy fitness functions with population-based evolutionary algorithms.

In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII, Aberystwyth, United Kingdom, January 17 - 20, 2015*, pages 62–68.

[Dantzig, 1963] Dantzig, G. (1963). *Linear programming and extensions*. Rand Corporation Research Study. Princeton Univ. Press, Princeton, NJ.

[de Oliveira and Sagastizàbal, 2014] de Oliveira, W. and Sagastizàbal, C. (2014). Bundle methods in the XXIst century: A bird's-eye view. *Pesquisa Operacional*, 34(3):647–670.

[Decock, 2014] Decock, J. (2014). *Hybridization of dynamic optimization methodologies*. Thesis, Université Paris Sud - Paris XI.

[Decock and Teytaud, 2013] Decock, J. and Teytaud, O. (2013). Noisy optimization complexity under locality assumption. In *Proceedings of the twelfth workshop on Foundations of genetic algorithms XII*, FOGA XII '13, pages 183–190, New York, NY, USA. ACM.

[Devroye et al., 1997] Devroye, L., Györfi, L., and Lugosi, G. (1997). *A Probabilistic Theory of Pattern Recognition*. Springer.

[Dong et al., 2011] Dong, C., Huang, G., Cai, Y., and Xu, Y. (2011). An interval-parameter minimax regret programming approach for power management systems planning under uncertainty. *Applied Energy*, 88(8):2835 – 2845.

[Dupač, 1957] Dupač, V. (1957). O Kiefer-Wolfowitzově aproximační Methodě. *Časopis pro pěstování matematiky*, 082(1):47–75.

[Dupač, 1958] Dupač, V. (1958). Notes on stochastic approximation methods. *Czechoslovak Mathematical Journal*, 08(1):139–149.

[Dvoretzky, 1956] Dvoretzky, A. (1956). On Stochastic Approximation. In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 39–55. University of California Press.

[Efron, 1982] Efron, B. (1982). *The Jackknife, the Bootstrap, and Other Resampling Plans*, volume 38. Society for Industrial and Applied Mathematics.

[Ernst, 2015a] Ernst, D. (2015a). Se déconnecter du réseau électrique peut être rentable, L'Echo. http://www.lecho.be/actualite/archive/Se_deconnecter_du_reseau_electrique_peut_etre_rentable.9696845-1802.art?ckc=1.

[Ernst, 2015b] Ernst, D. (2015b). The Global Grid for Empowering Renewable Energy - TEDx Liège. http://tedxtalks.ted.com/video/The-Global-Grid-for-Empowering.

[Fabian, 1967] Fabian, V. (1967). Stochastic approximation of minima with improved asymptotic speed. *The Annals of Mathematical Statistics*, 38(1):191–200.

[Fackle Fornius, 2008] Fackle Fornius, E. (2008). *Optimal Design of Experiments for the Quadratic Logistic Model*. PhD thesis, Stockholm University, Department of Statistics.

[Finck and Beyer, 2010] Finck, S. and Beyer, H.-G. (2010). Benchmarking SPSA on BBOB-2010 noisy function testbed. In *Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, Portland, Oregon, USA, July 7-11, 2010, Companion Material*, pages 1665–1672.

[Fitzpatrick and Grefenstette, 1988] Fitzpatrick, J. M. and Grefenstette, J. J. (1988). Genetic algorithms in noisy environments. *Machine Learning*, 3(2):101–120.

[Fonteneau, 2015] Fonteneau, R. (2015). From Energy To Research ... to Sustainable Energy? http://raphaelfonteneau.blogspot.fr/2014/11/from-energy-to-research-to-sustainable.html. Accessed: 2016-04-18.

[Fournier and Teytaud, 2010] Fournier, H. and Teytaud, O. (2010). Lower Bounds for Comparison Based Evolution Strategies using VC-dimension and Sign Patterns. *Algorithmica*, 59(3):387–408.

[Frangioni, 2006] Frangioni, A. (2006). Solving nonlinear single-unit commitment problems with ramping constraints. *Operations Research*, 54(4):767–775.

[Gagliolo and Schmidhuber, 2005] Gagliolo, M. and Schmidhuber, J. (2005). A Neural Network Model for Inter-problem Adaptive Online Time Allocation.

In Duch, W., Kacprzyk, J., Oja, E., and Zadrożny, S., editors, *Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005*, number 3697 in Lecture Notes in Computer Science, pages 7–12. Springer Berlin Heidelberg. DOI: 10.1007/11550907_2.

[Gagliolo and Schmidhuber, 2006] Gagliolo, M. and Schmidhuber, J. (2006). Learning dynamic algorithm portfolios. *Annals of Mathematics and Artificial Intelligence*, 47(3):295–328.

[Gaudel et al., 2011] Gaudel, R., Hoock, J.-B., Pérez, J., Sokolovska, N., and Teytaud, O. (2011). A principled method for exploiting opening books. In *Computers and Games*, pages 136–144. Springer.

[Gerencsér, 1999] Gerencsér, L. (1999). Convergence rate of moments in stochastic approximation with simultaneous perturbation gradient approximation and resetting. *IEEE Transactions on Automatic Control*, 44:894–905.

[Giraud, 2015] Giraud, G. (2015). How Dependent Is Output Growth From Primary Energy? `http://www.slideshare.net/PaulineTSP/lien-entre-le-pib-et-lnergie-par-gal-giraud-ads-20140306`. Accessed: 2016-04-18.

[Granichin, 2003] Granichin, O. N. (2003). Optimal Convergence Rate of the Randomized Algorithms of Stochastic Approximation in Arbitrary Noise. *Automation and Remote Control*, 64(2):252–262.

[Grigoriadis and Khachiyan, 1995] Grigoriadis, M. D. and Khachiyan, L. G. (1995). A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18(2):53–58.

[Hamadi, 2013] Hamadi, Y. (2013). *Combinatorial Search - From Algorithms to Systems*. Springer.

[Hammel and Bäck, 1994] Hammel, U. and Bäck, T. (1994). Evolution strategies on noisy functions. How to improve convergence properties. In Davidor, Y., Schwefel, H.-P., and Männer, R., editors, *Parallel Problem Solving from Nature — PPSN III: International Conference on Evolutionary Computation, Jerusalem, Israel, October 9–14, 1994 Proceedings*, pages 159–168. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Hansen et al., 2015] Hansen, N., Arnold, D. V., and Auger, A. (2015). Evolution Strategies. In Kacprzyk, J. and Pedrycz, W., editors, *Springer Handbook of Computational Intelligence*, pages 871–898. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Hansen et al., 2006] Hansen, N., Gemperle, F., Auger, A., and Koumoutsakos, P. (2006). When do heavy-tail distributions help? In *Parallel Problem Solving from Nature - PPSN IX, 9th International Conference, Reykjavik, Iceland, September 9-13, 2006, Procedings*, pages 62–71.

[Hansen et al., 2009] Hansen, N., Niederberger, S., Guzzella, L., and Koumoutsakos, P. (2009). A Method for Handling Uncertainty in Evolutionary Optimization with an Application to Feedback Control of Combustion. *IEEE Transactions on Evolutionary Computation*, 13(1):180–197.

[Hansen and Ostermeier, 2001] Hansen, N. and Ostermeier, A. (2001). Completely Derandomized Self-Adaptation in Evolution Strategies. *Evol. Comput.*, 9(2):159–195.

[Hansen and Ros, 2010] Hansen, N. and Ros, R. (2010). Benchmarking a weighted negative covariance matrix update on the BBOB-2010 noisy testbed. In *Genetic and Evolutionary Computation Conference, GECCO 2010, Proceedings, Portland, Oregon, USA, July 7-11, 2010, Companion Material*, pages 1681–1688.

[He and Yao, 2003] He, J. and Yao, X. (2003). Towards an analytic framework for analysing the computation time of evolutionary algorithms. *Artificial Intelligence*, 145(1–2):59–97.

[Hedman et al., 2011] Hedman, K. W., Oren, S. S., and O'Neill, R. P. (2011). A review of transmission switching and network topology optimization. In *2011 IEEE Power and Energy Society General Meeting*, pages 1–7.

[Hotelling, 1941] Hotelling, H. (1941). Experimental determination of the maximum of a function. *The Annals of Mathematical Statistics*, 12(1):20–45.

[International Energy Agency, 2014] International Energy Agency (2014). World Energy Outlook 2014 - Executive Summary. `http://www.iea.org/Textbase/npsum/WEO2014SUM.pdf`. Accessed: 2016-04-18.

[International Energy Agency, 2015] International Energy Agency (2015). CO_2 Emissions from fuel combustion - highlights. http://www.iea.org/publications/freepublications/publication/CO2EmissionsFromFuelCombustionHighlights2015.pdf. Accessed: 2016-04-18.

[International Renewable Energy Agency, 2012] International Renewable Energy Agency (2012). Renewable Energy Technologies: Cost Analysis Series. http://www.irena.org/documentdownloads/publications/re_technologies_cost_analysis-csp.pdf. Accessed: 2016-05-04.

[J. E. Kelley, 1960] J. E. Kelley, J. (1960). The cutting-plane method for solving convex programs. *Journal of the Society for Industrial & Applied Mathematics*, 8(4):703–712.

[Jamieson et al., 2012] Jamieson, K. G., Nowak, R., and Recht, B. (2012). Query complexity of derivative-free optimization. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 2672–2680. Curran Associates, Inc.

[Jebalia et al., 2011] Jebalia, M., Auger, A., and Hansen, N. (2011). Log-Linear Convergence and Divergence of the Scale-Invariant $(1+1)$-ES in Noisy Environments. *Algorithmica*, 59(3):425–460.

[Jin and Branke, 2005] Jin, Y. and Branke, J. (2005). Evolutionary optimization in uncertain environments-a survey. *Transacions on Evolutionary Computation*, 9(3):303–317.

[Kadioglu et al., 2011] Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., and Sellmann, M. (2011). Algorithm selection and scheduling. In *Principles and Practice of Constraint Programming - CP 2011 - 17th International Conference, CP 2011, Perugia, Italy, September 12-16, 2011. Proceedings*, pages 454–469.

[Karmarkar, 1984] Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395.

[Kezunovic et al., 2014] Kezunovic, M., Popovic, T., Gurrala, G., Dehghanian, P., Esmaeilian, A., and Tasdighi, M. (2014). Reliable Implementation of Robust Adaptive Topology Control. In *2014 47th Hawaii International Conference on System Sciences*, pages 2493–2502.

[Khuri et al., 2006] Khuri, A. I., Mukherjee, B., Sinha, B. K., and Ghosh, M. (2006). Design issues for generalized linear models: A review. *Statistical Science*, 21(3):376–399.

[Kiefer and Wolfowitz, 1952] Kiefer, J. and Wolfowitz, J. (1952). Stochastic estimation of the maximum of a regression function. *The Annals of Mathematical Statistics*, 23(3):462–466.

[Kleinman et al., 1999] Kleinman, N. L., Spall, J. C., and Naiman, D. Q. (1999). Simulation-based optimization with stochastic approximation using common random numbers. *Management Science*, 45(11):1570–1578.

[Kocsis and Szepesvári, 2006] Kocsis, L. and Szepesvári, C. (2006). Discounted-UCB. *2nd PASCAL Challenges Workshop*, pages 784–791.

[Kormushev and Caldwell, 2012] Kormushev, P. and Caldwell, D. G. (2012). Direct policy search reinforcement learning based on particle filtering. In *The 10th European Workshop on Reinforcement Learning (EWRL 2012), part of the International Conference on Machine Learning (ICML 2012)*, Edinburgh, UK.

[Kotthoff, 2014] Kotthoff, L. (2014). Algorithm selection for combinatorial search problems: A survey. *AI Magazine*, 35(3):48–60.

[Kötzing et al., 2015] Kötzing, T., Lissovoi, A., and Witt, C. (2015). (1+1) EA on Generalized Dynamic OneMax. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*, FOGA '15, pages 40–51, New York, NY, USA. ACM.

[Lai and Robbins, 1985] Lai, T. and Robbins, H. (1985). Asymptotically Efficient Adaptive Allocation Rules. *Advances in Applied Mathematics*, 6(1):4–22.

[LaTorre et al., 2010] LaTorre, A., Muelas, S., and Peña, J. M. (2010). Benchmarking a MOS-based algorithm on the BBOB-2010 noisy function testbed. In *GECCO (Companion)*, pages 1725–1730.

[Li et al., 2012] Li, M., Luh, P. B., Michel, L. D., Zhao, Q., and Luo, X. (2012). Corrective line switching with security constraints for the base and contingency cases. *IEEE Transactions on Power Systems*, 27(1):125–133.

[Liu, 2015] Liu, J. (2015). *Portfolio Methods in Uncertain Contexts*. Thesis, Université Paris-Saclay.

[Liu and Teytaud, 2014] Liu, J. and Teytaud, O. (2014). Meta online learning: experiments on a unit commitment problem. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, Bruges, Belgium.

[MacKay, 2009] MacKay, D. J. C. (2009). *Sustainable Energy - Without the Hot Air*. UIT Cambridge Ltd., 1st edition.

[Marchal, 2015] Marchal, D. (2015). Les enjeux de la Tansition énergétique pour les smart grids. `https://webcache.googleusercontent.com/search?q=cache:XhZL_GuBJmkJ:https://www.inria.fr/content/download/99541/1334708/version/2/file/PRESENTATION-DAVID-MARCHAL.pdf+&cd=3&hl=fr&ct=clnk&gl=fr&client=ubuntu`.

[Massé and Ollivier, 2015] Massé, P.-Y. and Ollivier, Y. (2015). Speed learning on the fly. Preprint.

[Miller, 1997] Miller, B. L. (1997). *Noise, Sampling, and Efficient Genetic Algorithms*. Thesis, University of Illinois at Urbana-Champaign.

[Miller and Goldberg, 1996] Miller, B. L. and Goldberg, D. E. (1996). Genetic Algorithms, Selection Schemes, and the Varying Effects of Noise. *Evolutionary Computation*, 4:113–131.

[Mnih et al., 2008] Mnih, V., Szepesvári, C., and Audibert, J.-Y. (2008). Empirical Bernstein Stopping. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 672–679, New York, NY, USA. ACM.

[Moré and Wild, 2009] Moré, J. J. and Wild, S. M. (2009). Benchmarking Derivative-Free Optimization Algorithms. *SIAM Journal on Optimization*, 20(1):172–191.

[Najafi and Pourjamal, 2012] Najafi, S. and Pourjamal, Y. (2012). A New Heuristic Algorithm for Unit Commitment Problem. *Energy Procedia*, 14:2005–2011. 2011 2nd International Conference on Advances in Energy Engineering (ICAEE).

[Nash, 1951] Nash, J. (1951). Non-cooperative games. *Annals of mathematics*, pages 286–295.

[Nash, 1952] Nash, J. (1952). Some games and machines for playing them. Technical Report D-1164, Rand Corporation.

[NERC, 2013] NERC (2013). Special reliability assessment: accomodating an increased dependence on natural gas for electric power phase ii: a vulnerability and scenario assessment for the north american bulk power system. Technical report, North American Electric Reliability Corporation.

[Nesterov, 2004] Nesterov, Y. (2004). *Introductory lectures on convex optimization. A basic course.* Boston: Kluwer Academic Publishers.

[Official Journal of the European Union, 2010] Official Journal of the European Union (2010). REGULATION (EU) No 994/2010 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 20 Octobre 2010 concerning measures to safeguard security of gas supply and repealing Council Directive 2004/67/EC. `http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=OJ:L:2010:295:FULL&from=EN`. Accessed: 2016-04-18.

[Ohlström et al., 2000] Ohlström, M. O., Lehtinen, K. E., Moisio, M., and Jokiniemi, J. K. (2000). Fine-particle emissions of energy production in finland. *Atmospheric Environment*, 34(22):3701 – 3711.

[Ong et al., 2003] Ong, Y. S., Lum, K., Nair, P. B., Shi, D., and Zhang, Z. (2003). Global convergence unconstrained and bound constrained surrogate-assisted evolutionary search in aerodynamic shape design. In *Congress on Evolutionary Computation, Special Session on Design Optimisation with Evolutionary Computation(CEC'03)*, pages 1856–1863, Canberra, Australia.

[Padhy, 2004] Padhy, N. P. (2004). Unit commitment-a bibliographical survey. *IEEE Transactions on Power Systems*, 19(2):1196–1205.

[Pereira and Pinto, 1991] Pereira, M. V. F. and Pinto, L. M. V. G. (1991). Multistage stochastic optimization applied to energy planning. *Math. Program.*, 52(2):359–375.

[Pinson, 2013] Pinson, P. (2013). Renewable Energy Forecasts Ought to be Probabilistic! WIPFOR seminar, EDF.

[Polyak and Tsybakov, 1990] Polyak, B. T. and Tsybakov, A. B. (1990). Optimal order of accuracy of search algorithms in stochastic optimization. *Problems of Information Transmission*, 26(2):126–133.

[Powell, 2004] Powell, M. J. D. (2004). The NEWUOA software for unconstrained optimization with derivatives. Technical Report NA2004/08, Department of Applied Mathematics and Theoretical Physics.

[Powell, 2008] Powell, M. J. D. (2008). Developments of NEWUOA for minimization without derivatives. *IMA Journal of Numerical Analysis*, 28(4):649–664.

[Powell, 2007] Powell, W. B. (2007). *Approximate Dynamic Programming : Solving the curse of dimensionality*. Wiley series in Probability and Statistics. Wiley-Interscience.

[Pulina and Tacchella, 2009] Pulina, L. and Tacchella, A. (2009). A Self-adaptive Multi-engine Solver for Quantified Boolean Formulas. *Constraints*, 14(1):80–116.

[Quenouille, 1949] Quenouille, M. H. (1949). Approximate tests of correlation in time-series. *Journal of the Royal Statistical Society. Series B (Methodological)*, 11(1):68–84.

[Ralf et al., 2014] Ralf, D., Elham, H., James, H., Anouk, H., Laura, E.-K., Simon, P., Howard, R., Jonathan, S., and Katja, Y. (2014). Reducing European Dependence on Russian Gas: distinguishing natural gas security from geopolitics. *Oxford Institute for Energy Studies*.

[Raso, 2013] Raso, L. (2013). *Optimal Control of Water Systems Under Forecast Uncertainty: Robust, Proactive, and Integrated*. PhD thesis, Delft University of Technology.

[Rechenberg, 1965] Rechenberg, I. (1965). Cybernetic solution path of an experimental problem. In *Royal Aircraft Establishment Translation No. 1122, B. F. Toms, Trans.* Ministry of Aviation, Royal Aircraft Establishment, Farnborough Hants.

[Rechenberg, 1973] Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog.

[Rechenberg, 1994] Rechenberg, I. (1994). *Evolutionsstrategie '94*. Frommann-Holzboog.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. (1951). A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, 22(3):400–407.

[Rolet and Teytaud, 2010a] Rolet, P. and Teytaud, O. (2010a). Adaptive noisy optimization. In Di Chio, C., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcazar, A. I., Goh, C.-K., Merelo, J. J., Neri, F., Preuß, M., Togelius, J., and Yannakakis, G. N., editors, *Applications of Evolutionary Computation*, volume 6024 of *Lecture Notes in Computer Science*, pages 592–601. Springer Berlin Heidelberg.

[Rolet and Teytaud, 2010b] Rolet, P. and Teytaud, O. (2010b). Bandit-Based Estimation of Distribution Algorithms for Noisy Optimization: Rigorous Runtime Analysis. In Blum, C. and Battiti, R., editors, *Learning and Intelligent Optimization: 4th International Conference, LION 4, Venice, Italy, January 18-22, 2010. Selected Papers*, pages 97–110. Springer Berlin Heidelberg, Berlin, Heidelberg.

[Ros, 2010a] Ros, R. (2010a). Black-box optimization benchmarking the IPOP-CMA-ES on the noisy testbed: comparison to the BIPOP-CMA-ES. In *GECCO (Companion)*, pages 1511–1518.

[Ros, 2010b] Ros, R. (2010b). Comparison of NEWUOA with different numbers of interpolation points on the BBOB noisy testbed. In *GECCO (Companion)*, pages 1495–1502.

[RTE-ft, 2014] RTE-ft (2014). RTE forecast team: Electricity Consumption in France: Characteristics and Forecast method. `http://clients.rte-france.com/lang/an/clients_producteurs/vie/courbes_methodologie.jsp`. Accessed: 2016-09-05.

[Saisirirat et al., 2013] Saisirirat, P., Chollacoop, N., Tongroon, M., Laoonual, Y., and Pongthanaisawan, J. (2013). Scenario Analysis of Electric Vehicle Technology Penetration in Thailand: Comparisons of Required Electricity with Power Development Plan and Projections of Fossil Fuel and Greenhouse Gas Reduction. *Energy Procedia*, 34(0):459 – 470. 10th Eco-Energy and Materials Science and Engineering Symposium.

[Samulowitz and Memisevic, 2007] Samulowitz, H. and Memisevic, R. (2007). Learning to Solve QBF. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, pages 255–260, Vancouver, British Columbia, Canada. AAAI Press.

[Saravanan et al., 2013] Saravanan, B., Das, S., Sikri, S., and Kothari, D. P. (2013). A solution to the unit commitment problem—a review. *Frontiers in Energy*, 7(2):223–236.

[Schein and Ungar, 2007] Schein, A. I. and Ungar, L. H. (2007). Active learning for logistic regression: an evaluation. *Machine Learning*, 68(3):235–265.

[Schmetterer, 1961] Schmetterer, L. (1961). Stochastic Approximation. In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics*, pages 587–609. University of California Press.

[Schoenauer and Ronald, 1994] Schoenauer, M. and Ronald, E. (1994). Neurogenetic truck backer-upper controller. *Proceedings of the First IEEE Conference on Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence*, 2:720–723.

[Scholz, 2007] Scholz, F. (2007). The Bootstrap Small Sample Properties. Technical report, Boeing Computer Services, Research and Technology.

[Schrijver, 1986] Schrijver, A. (1986). *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., New York, NY, USA.

[Schwefel, 1974] Schwefel, H.-P. (1974). Adaptive Mechanismen in der biologischen Evolution und ihr Einfluss auf die Evolutionsgeschwindigkeit. Interner Bericht der Arbeitsgruppe Bionik und Evolutionstechnik am Institut für Mess- und Regelungstechnik Re 215/3, Technische Universität Berlin.

[Shamir, 2013] Shamir, O. (2013). On the complexity of bandit and derivative-free stochastic convex optimization. In *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, pages 3–24.

[Shapiro, 2011] Shapiro, A. (2011). Analysis of stochastic dual dynamic programming method. *European Journal of Operational Research*, 209(1):63–72.

[Sheble and Fahd, 1994] Sheble, G. B. and Fahd, G. N. (1994). Unit commitment literature synopsis. *IEEE Transactions on Power Systems*, 9(1):128–135.

[Siqueira et al., 2006] Siqueira, T. G., Zambelli, M., Cicogna, M., Andrade, M., and Soares, S. (2006). Stochastic dynamic programming for long term hydrothermal scheduling considering different streamflow models. In *Probabilistic Methods Applied to Power Systems, 2006. PMAPS 2006. International Conference on*, pages 1–6.

[SolarPower Europe, 2015] SolarPower Europe (2015). The succesful stress test of Europe's power grid - more ahead. `http://www.solarpowereurope.org/fileadmin/user_upload/documents/Policy_Papers/entsoe_spe_pp_solar_eclipse_2015_web_FINAL.pdf`. Accessed: 2016-04-18.

[Spall, 1987] Spall, J. C. (1987). A stochastic approximation technique for generating maximum likelihood parameter estimates. In *Proceedings of the American Control Conference*, pages 1161–1167. IEEE Transactions on Automatic Control.

[Spall, 1992] Spall, J. C. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *Automatic Control, IEEE Transactions on*, 37(3):332–341.

[Spall, 2000] Spall, J. C. (2000). Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE Transactions on Automatic Control*, 45(10):1839–1853.

[Spall and Cristion, 1998] Spall, J. C. and Cristion, J. A. (1998). Model-free control of nonlinear stochastic systems with discrete-time measurements. *IEEE Transactions on Automatic Control*, 43(9):1198–1210.

[St-Pierre and Liu, 2014] St-Pierre, D. L. and Liu, J. (2014). Differential Evolution Algorithm Applied to Non-Stationary Bandit Problem. In *2014 IEEE Congress on Evolutionary Computation (IEEE CEC 2014)*, pages 2397–2403.

[St-Pierre and Teytaud, 2014] St-Pierre, D. L. and Teytaud, O. (2014). The Nash and the Bandit Approaches for Adversarial Portfolios. In *Proceedings of the international conference on Computational Intelligence in Games (CIG) 2014*, pages 1–7.

[Stern and Enflo, 2013] Stern, D. I. and Enflo, K. (2013). Causality between energy and output in the long-run. *Energy Economics*, 39:135 – 146.

[Storn and Price, 1995] Storn, R. and Price, K. (1995). Differential Evolution- A Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces. Technical report, CA, 1995, Tech. Rep. TR-95–012.

[Sutton and Barto, 1998] Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning - An Introduction*. MIT Press, Cambridge, MA, USA, 1st edition.

[SYSTINT Workgroup, 2007] SYSTINT Workgroup (2007). *European, CIS and Mediterranean Interconnection: State of Play 2006.* Ucte–Eurelectric.

[Tao Group, 2008] Tao Group (2008). Tao games axis. `http://www.lri.fr/~teytaud/games.html`. [Online; accessed 12-October-2016].

[Teytaud, 2008] Teytaud, O. (2008). When does quasi-random work ? In *Parallel Problem Solving from Nature – PPSN X: 10th International Conference, Dortmund, Germany, September 13-17, 2008. Proceedings*, pages 325–336, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Teytaud et al., 2005] Teytaud, O., Jebalia, M., and Auger, A. (2005). Algorithms (X,sigma,eta) : quasi-random mutations for Evolution Strategies. In *Evolution Artificielle*, pages 296–307, Lille, France. Springer (Artificial Evolution).

[The Telegraph, 2015] The Telegraph (2015). State of emergency declared in Crimea after activists 'blow up' electricity pylons. `http://www.telegraph.co.uk/news/worldnews/europe/ukraine/12010289/Crimea-plunged-into-darkness.html`. Accessed: 2016-04-21.

[Tran and Jin, 2010] Tran, T.-D. and Jin, G.-G. (2010). Benchmarking real-coded genetic algorithm on noisy black-box optimization testbed. In *GECCO (Companion)*, pages 1739–1744.

[United Nations, 2015] United Nations (2015). Adoption of the Paris Agreement. `http://unfccc.int/resource/docs/2015/cop21/eng/l09.pdf`. Accessed: 2016-04-18.

[Utgoff, 1989] Utgoff, P. E. (1989). Perceptron Trees: A Case Study in Hybrid Concept Representations. *Connection Science*, 1(4):377–391.

[Vaart and Wellner, 1996] Vaart, A. V. D. and Wellner, J. (1996). *Weak Convergence and Empirical Processes*. Springer series in statistics.

[Vapnik and Chervonenkis, 1968] Vapnik, V. and Chervonenkis, A. (1968). On the uniform convergence of frequencies of occurence events to their probabilities. *Soviet Mathematics-Doklady 9, 915-918*.

[Vapnik, 1995] Vapnik, V. N. (1995). *The Nature of Statistical Learning*. Springer Verlag.

[Vassena et al., 2003] Vassena, S., Mack, P., Rousseaux, P., Druet, C., and Wehenkel, L. (2003). A Probabilistic Approach to Power System Network Planning under Uncertainties. *IEEE Bologna Power Tech Conference Proceedings*, 2:6.

[Vassilevska et al., 2006] Vassilevska, V., Williams, R., and Woo, S. L. M. (2006). Confronting Hardness Using a Hybrid Approach. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 1–10, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics.

[Virmani et al., 1989] Virmani, S., Adrian, E. C., Imhof, K., and Mukherjee, S. (1989). Implementation of a Lagrangian relaxation based unit commitment problem. *Power Systems, IEEE Transactions on*, 4(4):1373–1380.

[von Stengel, 2002] von Stengel, B. (2002). Computing equilibria for two-person games. *Handbook of Game Theory*, 3:1723 – 1759.

[Wellner, 2014] Wellner, J. (2014). Bootstrap and Jackknife Estimation of Sampling Distributions.

[Whitley, 2015] Whitley, D. (2015). Mk Landscapes, NK Landscapes, MAX-kSAT: A Proof That the Only Challenging Problems Are Deceptive. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 927–934, New York, NY, USA. ACM.

[Whitmarsh et al., 2012] Whitmarsh, A., Bojanowski, S., and Barber, W. (2012). Gas security of supply report. Technical report, Ofgem.

[Wolpert and Macready, 1997] Wolpert, D. H. and Macready, W. G. (1997). No Free Lunch Theorems for Optimization. *Trans. Evol. Comp*, 1(1):67–82.

[World Health Organization, 2015] World Health Organization (2015). World Health Assembly closes, passing resolutions on air pollution and epilepsy. `http://www.who.int/mediacentre/news/releases/2015/wha-26-may-2015/en/`. Accessed: 2016-04-18.

[Xu et al., 2008] Xu, L., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2008). SATzilla: portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research*, 32:565–606.

[Xu et al., 2011] Xu, L., Hutter, F., Hoos, H. H., and Leyton-Brown, K. (2011). Hydra-MIP: Automated algorithm configuration and selection for mixed integer programming. In *RCRA workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion at the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 16–30.

[Yu, 1994] Yu, B. (1994). Rates of convergence for empirical processes of stationary mixing sequences. *Ann. Probab.*, 22(1):94–116.

[Zambelli et al., 2011] Zambelli, M., Soares Filho, S., Toscano, A., Santos, E. d., and Silva Filho, D. d. (2011). NEWAVE versus ODIN: comparison of stochastic and deterministic models for the long term hydropower scheduling of the interconnected brazilian system. *Sba: Sociedade Brasileira de Automatica*, 22:598 – 609.

[Zaoui et al., 2005] Zaoui, F., Fliscounakis, S., and Gonzalez, R. (2005). Coupling OPF and topology optimization for security purposes. In 15th *Power Systems Computation Conference*, pages 22–26.

[Zedek, 1965] Zedek, M. (1965). Continuity and Location of Zeros of Linear Combinations of Polynomials. *Proceedings of the American Mathematical Society*, 16(1):78–84.

[Zhan, 2005] Zhan, X. (2005). Extremal eigenvalues of real symmetric matrices with entries in an interval. *SIAM J. Matrix Analysis Applications*, 27(3):851–860.

[Zhou et al., 2004] Zhou, Z., Ong, Y. S., and Nair, P. B. (2004). Hierarchical surrogate-assisted evolutionary optimization framework. In *IEEE Congress on Evolutionary Computation 2004*, volume 2, pages 1586–1593.