Contributions to Convergence Analysis of Noisy Optimization Algorithms

Sandra Astete Morales Astete Morales

NNT : 2016SACLS327

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. Sandra Astete-Morales

Contributions to Convergence Analysis of Noisy Optimization Algorithms

Thèse présentée et soutenue à Gif-sur-Yvette, le 05 octobre 2016.

Composition du Jury :

M.	Benjamin Doerr	Professeur Ecole Polytechnique	(Président du jury)
M.	Timo Kotzing	Professeur Hasso-Plattner-Institut	(Rapporteur)
M.	Adam Prugel-Bennett	Professeur University of South-Hampton	(Rapporteur)
Mme	Coralia Cartis	Professeur University of Oxford	(Examinatrice)
M.	Per Kristian Lehre	Professeur University of Nottingham	(Examinateur)
M.	Jonathan Shapiro	Professeur University of Manchester	(Examinateur)
M.	Olivier Teytaud	Directeur de recherche INRIA Saclay TAO	(Directeur de thèse)
M.	Marc Schoenauer	Directeur de recherche INRIA Saclay TAO	(Co-Encadrant)

université PARIS-SACLAY
ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Contributions à l'Analyse de Convergence d'Algorithmes d'Optimisation Bruitée

Mots clefs : Optimisation, Optimization Bruitée, Analyse d'algorithmes, Convergence des algorithmes d'optimisation

Résumé : Cette thèse montre plusieurs contributions à l'analyse de convergence d'algorithmes pour l'optimisation bruitée. Nous analysons les taux de convergence pour deux algorithms de different type. Le premier est un algorithme de type *linesearch* et le deuxieme de type Randomized Search Heuristic. Nous prouvons que l'algorithme en utilisant une approximation de la matrice Hessiene peux atteindre les memes taux de converrgence que d'autres algorithmes optimaux, quand les parametres sont bien ajustes. Par ailleurs, nous analysons l'ordre de convergence pour les Evolution Stratagies pour optimisation bruitee, en utilisant des reevaluations. Nous obtenons des resultats theoriques et experimentaux pour la convergence log-log. Aussi, nous prouvons une borne inferieure pour le taux de convergence des Evolution Strategies. Nous extendons le travail et appliquons les memes schema de reevaluation pour le cas discret qui present des perturbations bruitees dans la fonction objective. Finalement, nous analysons la mesure de performance en elle meme, en utilisant une comparaison entre deux indicateur de qualite sur differents algorithmes. Nous prouvons que l'utilisation d'un indicateaur inapproprie nous ammene a des resultats misleading quand nous comparons plusieurs algorithmes d'optimisation bruitee.

Title : Contributions to Convergence Analysis of Noisy Optimization Algorithms

Keywords : Optimization, Noisy Optimization, Algorithm Analysis, Convergence of Optimization Algorithms

Abstract : This thesis shows several contributions to the convergence analysis for noisy optimization algorithms. We analyze the convergence rates for two algorithms of different type. One of the type linesearch and the other of the type Randomized Search Heuristics. We prove that the algorithm using an approximation of the Hessian matrix can reach the same rates as other optimal algorithms, when the parameters are well adjusted. Also, we analyze the convergence order for Evolution Strategies for noisy optimization, using reevaluation. We obtain theoretical an empirical results for log-log convergence. We also prove a lower bound for the convergence rate of Evolution Strategies. We extend the work to the application of similar reevaluation schemes to a discrete case presenting noisy perturbations in the objective function. Finally, we analyze the measure of performance itself by comparing two quality indicator over different algorithms. We prove that the use of an inadequate indicator can lead to misleading results when comparing several noisy optimization algorithms.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France

Synthèse

Cette thèse présente plusieurs contributions à l'analyse de convergence d'algorithmes pour l'optimisation bruitée.
Nous commençons par introduire le problème analysé ici : maximiser ou minimiser une fonction corrompue par le bruit. Cela veut dire que les observations des évaluations de points ne sont pas précises : nous pouvons obtenir deux résultats différents si nous évaluons deux fois le même point. Dans cette situation, les algorithmes d'optimisation "traditionnels" (ou déterministes) ne peuvent pas garantir un résultat optimal. Nous analysons dans cette thèse différents aspects de ces algorithmes. En particulier, nous allons parler de différents processus a l'intérieur des algorithmes. En général, tous les algorithmes d'optimisation ont (au moins) deux types de processus : recherche et recommandation.
Il existe différentes manières de modifier les algorithmes déterministes pour pouvoir faire face à des problèmes bruitées. Dans cette thèse nous nous concentrons sur une méthode spécifique : la réévaluation. Avec cette méthode un même point est réévalué de nombreuses fois, de manière à obtenir une approximation proche de la valeur réelle de la fonction à ce point spécifique.
De plus, pour analyser la convergence des algorithmes il faut définir le type d'erreur que nous allons considérer pour déterminer si une recommandation de l'algorithme est "bonne". Dans cette thèse nous regarderons en détail le "Simple Regret". Cette erreur mesure la distance entre la valeur de la fonction objectif au point recommandé et la valeur de l'optimum.
Les contributions de cette thèse à l'analyse d'algorithmes d'optimisation bruitée sont réparties sur plusieurs axes. Le chapitre 4 propose une analyse d'un algorithme de type "Linesearch". Un example célèbre de ce type d'algorithme est la méthode de descente de gradient. Dans cette thèse nous analysons un algorithme général de type "Linesearch". Cet algorithme général inclut plusieurs éléments : une distribution pour la recherche, une méthode de recommandation et des paramètres adaptatifs. Donc, avec cet algorithme général, en ajustant correctement les paramètres, nous pouvons obtenir la méthode de descente de gradient et l'algorithme de Newton. Nous analysons ensuite la convergence de l'algorithme de Newton. La méthode de recommandation dans ce cas utilise un grand nombre de réévaluations pour obtenir une estimation de la vrai valeur de l'évaluation. Les paramètres de "step-size" et du nombre de réévaluations s'adaptent en fonction du nombre d'itérations. Nous prouvons que l'algorithme de Newton pour l'optimisation bruitée atteint le taux de convergence de plusieurs algorithmes de la littérature, lorsque les paramètres sont ajustés de manière adéquate. Nous analysons ensuite la convergence d'un autrs paramètres. Nous analysons ensuite la convergence de e type d'algorithme : "Evolution Strategy". Nous analysons trois types de réévaluations : exponentiel, adaptatif et polynômial dans le nombre d'itérations. Nous assumons que l'algorithme "Evolution Strategy" converge dans le cas déterministe. Nous prouvons que l'algorithme converge aussi dans le cas bruité, mais avec un taux de convergence plus lent, en utilisant la réévaluation exponentielle et adaptative. Nous montrons aussi des résultats expérimentaux pour la convergence de la réévaluation polynômiale.
En poursuivant l'analyse des algorithmes "Evolution Strategy", nous analysons le taux de convergence de manière plus détaillée. Nous savons déjà que ce taux est plus lent que dans le cas déterministe, mais ne savons pas encore le taux exacte. Nous prouvons donc une borne inférieur pour le taux de convergence de l'algorithme "Evolution Strategy". Donc, nous savons maintenant que l'algorithme ne peut pas atteindre le taux de convergence optimal pour

l'optimisation de fonctions quadratiques bruitées. Il est important de noter que la preuve assume un type spécial de recherche de points à l'intérieur de l'algorithme. La recherche doit être contenu dans le voisinage du point de recommandation actuelle.
Nous analysons ensuite la mesure de performance de manière générique. Dans les expériences réelles, il n'est parfois pas possible de calculer le "Simple Regret", spécialement dans le cas de l'optimisation bruitée. Donc, afin d'évaluer la performance d'un algorithme nous devons faire appel à des approximations de la mesure "Simple Regret". Nous analysons le taux de convergence de plusieurs algorithmes en utilisant le "Simple Regret" et une approximation du "Simple Regret". Nous prouvons mathématiquement et par expériences que les résultats peuvent être contradictoires en fonction de la mesure utilisée.
Le dernier résultats de cette thèse est une analyse dans le cas des problèmes d'optimisation discrète. Nous utilisons les mêmes techniques de réévaluation pour prouver que nous pouvons adapter les algorithmes classiques d'optimisation discrète déterministe à la résolution des problèmes bruitées. Nous obtenons les temps de fonctionnement (*runtime*) des algorithmes en fonctions du nombre d'évaluations.

Acknowledgments

I would like to express my sincere gratitude to my advisor Olivier Teytaud for offering me the opportunity to do a PhD under his supervision. He is an advocate of freedom in research and was supportive of my decisions. He was especially concerned of my formation in the beginning of my research career, trying to fill the doubts I had and ensuring I was, in general, doing a good job. I also have to mention the participation of my co-advisor, March Schoenauer, who was an incredible support in the last months of my PhD. He had to deal with all the problems related to the end of a thesis and he did it with extreme commitment. I appreciate very much all of his advices and I admire the way he makes things happen. Both Olivier and Marc have taught me so much and I will always be grateful to them.

I thank warmly to both of my reviewers: Timo Kotzing and Adam Prugel-Bennet. They made an admirable work by reading carefully all my thesis. Thanks to them, my manuscript is more clear. I wish the process of revision would have allow me to have conversations with them and make so many more improvements. Thanks also to all the other members of my jury: Benjamin Doerr, Per Kristian Lehre and Jonathan Shapiro for all their questions and comments. Special thanks to Coralia Cartis, who made very insightful and fruitful remarks during the PhD defense.

I was lucky enough to land at TAO, a very fertile research group lead by two great researchers: Marc and Michele. I learnt countless lessons during my stay there. A mention to Anne Auger, who was kind enough to discuss with me about her opinions in research. Thanks to this experience I will always keep in mind how academic research is trying to have a positive influence in the development of science. But my experiences was not only scientific, but also about researchers and research institutions in general. I had wonderful times with my colleagues in the every day life and in extra-curricular activities. I wish everybody a bright future in whatever destiny you choose. Special thanks to my PhD colleagues and co-authors Marie-Liesse Cauwet and Jialin Liu. And of course to all the other members of the team Nicolas, Asma, Ouassim, Riad, Antoine, Gaetan, Adrien, Jeremy and so many others!

One thing that I appreciate very much from my PhD at INRIA was the help from all the administrative people related to me in some way or another. It is definitively not easy to do all the procedures necessary as a foreigner. So I start by mentioning Valerie Berthou, who saved me many times during the jungle of administrative papers in France. Also Stephanie Druetta, who also helped me countless times when I was lost. Finally, Olga Mwana Mobulakani, who was always willing to guide me, even when it was out of her scope. This is of course a non-exhaustive list, in general all the administrative personal in INRIA is deeply committed and doing a great job. I always appreciated the excellent job they do and how they are also always ready to make an extra effort.

I have to thank my family and all my friends, who helped me stay sane during this adventure. Mención especial a mis queridos amigos del colegio: Silvana, Javiera, Consuelo, Beatriz, Daniela, Mono, Natham, Catalina, Alejandra, Gunther y tantos otros (siempre destacando a mi club de

Lulú, incorruptible y eterno!). A mi amigo personal Inti, que ha sido constante y estado presente durante años de años. A los amigos hechos acá: María Jesus, David, Laura, Despi, Felipe, Matías, Daniel, Farida, Louisette y más. A los colegas y amigos de la Escuela, primero a mi querida amiga Polaka, a quien tuve la suerte de conocer desde el Bachillerato y continuar siendo amigas durante todos nuestros años en la Universidad. Una mención especial a todos mis muy queridos amigos DIM y particularmente a aquellos que me ayudaron con sus consejos durante la tesis: Pedro, Pita, Johan, Valentina, Emilio y Riffo (quien celebra su simposio de 30 años en 2017). A los grandes profesores que tuve durante mi carrera: Cristián Reyes, Servet Martínez, Nancy Lacourly, Manuel del Pino, Jaime San Martín y muchos otros que merecen ser nombrados. Y por supuesto, a toda mi hermosa familia que ha sido un apoyo fundamental, siempre animándome a alcanzar otros horizontes y entregándome todo el cariño necesario a la distancia. Mi padre, mis tíos, mis primos, Manolo, Tito y por sobre todo mi hermano Cristobalito y a Helga, la super mujer que tengo de mamá (y el Marlon!). Y nunca olvidar a mi Carlotita, alias abuela Raquel, que habría disfrutado al máximo cada uno de los detalles de mis viajes y experiencias.

Merci aussi à la famille de Jean-Marc qui a été très acueillante du début de mon séjour en France. Y finalmente gracias a Jean-Marc, que ha sido un compañero indispensable. Siempre con una sonrisa, siempre mostrándome el lado positivo y siempre conmigo. Gracias por estar y por creer en mí. Ya vendrá el día en que pueda comprobar la fama de los poetas chilenos y escribir un poema exclusivo para ti. Por ahora al menos mis compatriotas me ayudan y una palabra entonces, una sonrisa bastan.

Contents

Part I

Introduction and State of the Art

Chapter 1

Introduction

Noisy Optimization is the domain that studies the minimization or maximization of functions corrupted by noise. In other words, Noisy Optimization is the search for the optimum of a function that is hidden under realizations of a random variable. For instance, suppose we want to optimize $F(x) = x^2$. But each time the optimization method queries $F(x)$ for some specific x, it obtains $F(x) + \omega$, where ω is a realization of a random variable. Therefore, when we query the function value of an specific x, we have no certainty that this is in fact the "real" function value of x.

The advantage of the use of noisy functions is the resemblance to real-life optimization. Every time we solve an optimization problem using physical measures or computer simulated data we observe a component of error. Whether it is the precision of the instruments or an error in a model, we can represent this situation by assuming we are optimizing a *noisy function*. The disadvantage of the use of noisy functions is that the performance analysis of the optimization methods is more complex than using deterministic functions.

In general, an optimization algorithm (for deterministic or noisy functions) is composed of two fundamental methods or processes. First, the way to generate *search* points and second, the way to generate *recommendation* points (or solutions). In some cases search and recommendation points are the same. In other cases the search points are used to *explore* the domain, not necessarily around the optimum. We will see in this thesis that the acknowledgement or neglect of the difference between search and recommendation points can change drastically the performance of an algorithm, especially in noisy optimization algorithms.

In deterministic optimization, the most classic optimization method for differentiable functions is the Gradient Descent method. It uses the simple and powerful idea to follow the direction of the "greatest descent" at each point of the iteration. In the literature, the algorithms using this idea are called *Linesearch Methods*. The simple Gradient Descent method ensures that the next search point will have lower or equal function value than the current solution. And it makes no difference between search and recommendation point: at each iteration it generates only one point that is the approximated solution. The algorithm iterates moving in the direction of the negative gradient using a *step-size*. The step-size can be a fix or variable parameter. The use of the Hessian in the step-size gives rise to the Newton Method.

The performance analysis of the previous optimization methods uses the characteristics of the functions they can optimize. Even more, for the domain of Convex Optimization, the methods are created exactly by using the properties of convex functions. For the gradient descent the theory can ensure local convergence to a stationary point of the objective function, provided that the function and its derivative are smooth. While for the Newton method there is local convergence

with faster rate than the Gradient Descent (quadratic instead of linear), on a smaller convergence region. Since the Newton method uses the Hessian, it requires more from the objective function: smoothness of the second derivative of the function. Adding more properties, as convexity gives the analysis important advantages. For instance the first order condition is sufficient to have an optimum, which is not the case for only differentiable functions. The properties of the functions translate into fast convergence rates and the possibility to attain global convergence. And so forth, methods prepared to handle convex, and strongly convex functions have better convergence results, by being engineered to solve specific classes of functions. It is a natural consequence: as long as we have more information on the function, we can develop better algorithms to solve the optimization problem faster and more accurately.

But the high demand for properties of the functions, such as convexity or strong convexity, can be hard to meet in the optimization of real-world functions. This motivates the use of general purpose algorithms such as Random Search Heuristics. They are algorithms for global convergence with very few requirements for the objective function. In opposition to the classic optimization algorithms, Random Search Heuristics are created to emulate "natural" processes that lead to optimization in some sense (for instance: emulate evolution of a population). Evolutionary Algorithms, Ant Colony and Simulated Annealing are a few popular examples. The Random Search Heuristics share some common principles. They have some search method that usually includes a random component, therefore the search is guided by some *search distribution*. And they have some method to generate the recommendations or solutions, which may also include a random component.

The remarkable success of Random Search Heuristics in practice are unfortunately not always followed by a theoretical backup. The generality of the functions they can optimize is a handicap for their performance analysis. Even though the global convergence can be ensured in limited time in many cases, the convergence rates are not realistic. For instance [Droste *et al.*, 2002] proved the existence of functions for which the $(1+1)$- EA finds the global optimum in expected time $\Theta(d^d)$ where d is the dimension of the problem.

In 1952 Robbins and Monro started formally the analysis of Stochastic Optimization algorithms (see their original work in [Robbins and Monro, 1951]). Their algorithm is made to find the solution of a root problem, only having access to noisy function evaluations. This translates to the use of that algorithm to optimize a differentiable function that will have an optimum with derivative equal to zero. More algorithms were developed following the same idea: use finite differences to approximate the "real" gradient and use the direction of the greatest descent. Therefore they can be considered the "noisy counterpart" of the Linesearch methods in classical optimization.

The optimization of noisy functions can be done also using Random Search Heuristics. They are naturally robust in the presence of noise, because of their stochastic components. The added randomness allows for the algorithms to not stagnate in false optimums and explore the search space, even though, sometimes Random Search Heuristics need some modification in order to deal with the noise on the objective function. Notably the use of a reevaluation scheme is highly recommended in order to reduce the variance of the estimate of the real function evaluation. Even

though Random Search Heuristic such as Evolutionary Algorithms are used in practice for the resolution of noisy problems, the theoretical analysis of their performance is a young field and still developing.

The analysis of algorithms for noisy optimization problems shares some difficulties, no matter the type of algorithm we are analyzing. For instance, there is a limit on the improvement of the statistical error when we consider independent noise: the error decreases at rate $O(1/\sqrt{n})$ where n is the number of function evaluations. This represents a fundamental limit for any noisy optimization algorithm.

Another difficulty lies in the way to measure the *accuracy* of the solution output by the optimum. The most natural way to measure the precision or accuracy of a solution is to measure a posteriori the distance between the recommendation and the optimum. The measurement can be done either in the search space or in the image space. When we measure in the image space we use the difference $F(\hat{x}_t) - F(x^*)$ where $\hat{x}_t$ is the recommendation at time t and x^* is the real optimum. We call this difference *Simple Regret*. Notice that when we are in the deterministic case we can design a stop mechanism to test the algorithms, by stopping the optimization process once we reach a certain threshold for the Simple Regret. These measurements are meant to test the algorithm, and in real-life optimization it is harder to design automated stop mechanism in deterministic optimization. When we add noise to the function evaluations, it becomes an even more difficult task.

In this thesis we will study convergence rates for algorithms for Noisy Optimization. We will propose general frameworks and analyze their extent and performance. We will also discuss performance measures for Noisy Optimization algorithms. We will show the importance of the precise definition of a performance measure. We will provide theoretical analysis and empirical verification of our results.

The remainder of this chapter is an overview of the different topics explored in this thesis. The thesis is divided in three parts: Part I: Introduction and State of the Art, Part II: Contributions and Part III: Conclusions.

1.1 State of the Art

The State of the Art is devoted to the presentation of general optimization algorithms mentioned in the literature of Continuous (Deterministic and Stochastic[1]) Optimization. We describe how the Gradient Descent Method works and the main convergence results for Linesearch Methods. We discuss briefly the Trust Region Methods.

Then we move to the presentation of Stochastic Optimization. The development of algorithms specialized to handle noisy function evaluations is the center of this part. We present the main algorithms used in this thesis and the associated convergence results.

The last part of the State of the Art refers to the study of general purpose algorithms: the Random

[1] We will use interchangeably the terms "Stochastic Optimization" and "Noisy Optimization". We try to maintain the terms used by different authors so that the reader may find the original term used in the references.

Search Heuristics. These algorithms are versatile and can be used in discrete or continuous setting, deterministic or noisy. We present the relevant algorithms used in this thesis and the convergence results in the literature.

1.2 Contributions

We study noisy optimization algorithms from several angles. We analyze a Linesearch method that uses a second order estimation in Chapter 4. We obtain its convergence rates and prove that depending on the parametrization, the algorithm can reach convergence rates found in the literature. We also analyze the convergence of Evolution Strategies (considered as a Random Search Heuristic) when facing a noisy function in Chapters 5 and 6. In Chapter 5 we prove the convergence order. We give a lower bound for the convergence rate in Chapter 6. We investigate in Chapter 7 the realization of the theoretical convergence rates in practice. We observe the way to evaluate the performance of algorithms for noisy optimization and analyze the precision measures used for this evaluation. We conclude that the definition of the precision measure needs to be carefully designed in order to obtain a "correct" evaluation of the performance of the algorithms. Otherwise we can obtain an under- or overestimation of the performance of the algorithm. We present finally a contribution in a discrete setting in Chapter 8. We use the techniques for continuous functions from previous chapters in the discrete optimization case by generalizing the form of the algorithms.

1.2.1 Convergence Rates for General Noisy Optimization Algorithm

We start in Chapter 4 by proposing a general iterative noisy optimization algorithm and a property called "Low Square Error" (LSE). The general algorithm includes several elements: a distribution search, a method that outputs recommendations, and adaptive stepsizes and reevaluation number. The distribution search can be updated at each iteration depending on a step-size and the information of previous iterations. The recommendation method uses the reevaluation number to obtain estimates of the real function value of the search points. The stepsize and the reevaluation number are updated according to some parameters and the iteration number. The LSE property accounts for the relationship between search points and recommendation points. We prove that two types of Linesearch algorithms fit the framework and verify the LSE property: gradient and Hessian based algorithm. Then we prove convergence rates in terms of Simple and Cumulative Regret for the Hessian based algorithm. These rates are verified for any noisy function with quadratic expectation. We are able to reach the same convergence rates as in [Fabian, 1967; Shamir, 2013; Dupac, 1957; Rolet and Teytaud, 2010a], as well as confirm the conjecture from [Jebalia and Auger, 2008]. We obtain the results with only one algorithm, by varying its parameters in order to find the best convergence rates possible.

Indication of source: The content of Chapter 4 is based on the publication in Theoretical Computer Science 2016 under the name "Simple and Cumulative Regret for Continuous Noisy Optimization" (see [Astete-Morales *et al.*, 2016a]).

1.2.2 Log-log Convergence for Evolution Strategies

We turn now to the analysis of a special type of Random Search Algorithms on continuous domain: the Evolution Strategies. In Chapter 5 we analyze three reevaluation schemes used in Evolution Strategies with noisy functions. They are exponential, adaptive and polynomial in the iteration number. We use as an assumption that Evolution Strategies converge in the noise-free case for the sphere. [Auger, 2005] proves that the convergence is log-linear (given that the Evolution Strategy converges). We prove that in the case of Evolution Strategies with reevaluation exponential and adaptive, the convergence is log-log. We show with experimental results the performance of the Evolution Strategy with polynomial reevaluation scheme.

Indication of source: The content of Chapter 5 is based on the publication in Lecture Notes of Computer Science 2014 under the name "Log-log Convergence for Noisy Optimization" (see [Astete-Morales *et al.*, 2014]).

1.2.3 Lower bound on the Convergence Rate of ES

The performance of Evolution Strategies over the optimization of noisy functions is studied in Chapter 5 and it yields a convergence rate of linear order in the log-log scale. Nonetheless, there is no proof of the exact convergence rate that Evolution Strategies can reach. We prove in Chapter 6 a lower bound for the convergence rate of Evolution Strategies. For that we use the assumption that the distribution search does not sample points too far away from the optimum. The latter is represented by assuming that the stepsize grows as the square of the distance to the optimum. The assumption is verified by scale-invariant algorithms and it can also be verified *a posteriori* on experimental setup. Notably, an algorithm that does not verify the assumption is any algorithm that samples far away from the optimum.

We obtain that the convergence rate for Evolution Strategies sampling close to the optimum is no better than $\Omega(n^{-1/2})$. We also exhibit the theoretical result in experiments using a linesearch algorithm from [Shamir, 2013] and two ES: $(1+1)$- ES with polynomial and exponential reevaluation scheme (as in Chapter 5) and UHCMAES from [Hansen *et al.*, 2009].

Indication of source: The content of Chapter 6 is based on the publication in Proceedings of Foundations Of Genetic Algorithms 2015 under the name "Evolution Strategies with Additive Noise: A Convergence Rate Lower Bound" (see [Astete-Morales *et al.*, 2015]).

1.2.4 Performance Measure of Noisy Optimization Algorithms

The previous chapters show convergence rates of algorithms in order to solve the optimization problem. And we only use previously the same types of performance measures: Simple and Cumulative Regret. But the performance evaluation of algorithms in real-life is done using testbeds that simulate real situations. Using a range of types of functions and specific accuracy measures. We show in Chapter 7 that the use of an erroneous approximation of a performance measure can have misleading results. We analyze the differences between Simple Regret and two approximations of Simple Regret. The approximations are deceptive because they cannot reproduce the same

performance evaluation as Simple Regret. We find that the approximation can underestimate some algorithms and overestimate others. Therefore the analysis of the performance measure of testbeds is essential and needs to be carefully chosen in order to reflect the effects of the theoretical analysis

Indication of source: The content of Chapter 7 will be published in the Proceeding of Genetic and Evolutionary Computation Conference 2016 under the name "Analysis of Different Types of Regret in Continuous Noisy Optimization " (see [Astete-Morales *et al.*, 2016b]).

1.2.5 Convergence Rates using Reevaluation in Discrete Noisy Optimization

The use of reevaluation schemes is not limited to the optimization of functions on continuous domains. The analysis can be extended to algorithms in discrete settings. We present in Chapter 8 a general result considering noise represented by a continuous random variable (Gaussian or with bounded variance). Assuming we use an algorithm that solves a discrete problem, we propose a modification of the algorithm that maintains its convergence but now on the noisy counterpart of the problem.

We develop two reevaluation schemes. The first one assumes as known the runtime of the algorithm on the noise-free version of the objective function. The second one omits the runtime in the noise-free case and updates the reevaluation number depending on the number of iterations.

We prove that the convergence rate is modified by an extra logarithmic number of evaluations of the objective function in the case of Gaussian noise. In the case of bounded variance noise, the final runtime is quadratic in function of the runtime in the noise-free case.

Indication of source: The content of Chapter 8 is based on the publication in journal Theoretical Computer Science 2015 under the name "Analysis of runtime of optimization algorithms for noisy functions over discrete codomains" (see [Akimoto *et al.*, 2015]).

1.3 Disclaimer

I had the pleasure to collaborate with other researchers for all the articles included in this thesis. All of the joint work is done by a group effort and all the members of the groups are essential for the final result of each article. I have participated in the theoretical and empirical analysis of all the papers exhibited in this thesis and I am the author of all the figures in this thesis. I mention first the collaboration and guidance of Olivier Teytaud in all articles and Youhei Akimoto in [Akimoto *et al.*, 2015]. I would like to point out the participation of my fellow phd students Marie-Liesse Cauwet and Jialin Liu in the analysis of [Astete-Morales *et al.*, 2016a], especially in the proof in Section 4.3.1, included in the appendix of this thesis. The work of Jialin Liu in the paper [Astete-Morales *et al.*, 2014] was appreciable specially in the experimental part. The contribution of Marie-Liesse Cauwet in the theoretical analysis of [Astete-Morales *et al.*, 2015] and [Astete-Morales *et al.*, 2016b] was significant, specially the proof of Theorem 7.1, included in the appendix of this thesis. Finally I would like to note that all the authors of the papers included in this thesis are listed alphabetically in the official publications.

Chapter 2

Preliminaries

2.1 Basic Notation

We denote the positive integers by $\mathbb{N}$, the reals by $\mathbb{R}$ and the positive reals by $\mathbb{R}_+$. We denote by ω a random variable with the following precise definition.

$$\begin{aligned} \omega : \Theta &\to \mathbb{R} \\ \theta &\mapsto \omega(\theta), \end{aligned}$$

with distribution D_ω such that for any $a \in \mathbb{R}$, $D_\omega(a) = \mathbb{P}(\{\theta \in \Theta : \omega(\theta) \leq a\})$, where Θ is a sample space. Typically we will omit the dependence in θ. We denote by $\mathbb{E}_\omega$ the expectation with regard to the random variable ω. When it is not necessary, we will omit in the notation the dependency on ω and only use $\mathbb{E}$. $[a, b]$ will denote the interval between a and b. It can also denote the set $\{a, a+1, \ldots, b\}$ when we treat with discrete functions (Chapter 8). $\mathscr{C}_L^{k,p}(\mathcal{D})$ represents the class of functions k times continuously differentiable on $\mathcal{D}$, with p-th derivate Lipschitz with constant L.

2.2 Optimization Problem and Performance of Numerical Methods

Let x be a d-dimensional real vector: $x = (x^{(1)}, \ldots, x^{(d)})$, a set $\mathcal{D} \in \mathbb{R}^d$ and F some real valued function of x. Then, we search for

$$\min_{x \in \mathcal{D}} F(x). \tag{2.1}$$

We will call our minimization problem $\mathcal{P}$. Assume that we want to find the best method to solve $\mathcal{P}$. This means that we want to find a method that solves some *class* or *family* of problems, say $\mathcal{F}$, where $\mathcal{P}$ is a member of the class. Since we want to measure the performance of the algorithm on a family, we need to take that into account that we have only the information of the family $\mathcal{F}$ when we evaluate the performance of an algorithm. We introduce the concept of *oracle* $\mathcal{O}$. The oracle is a machine that answers the successive calls of the method. We will consider throughout this thesis an oracle of *zero order*: the oracle has access to the function evaluation of any point $x \in \mathcal{D}$. We could also consider oracles of *first order* and *second order*, which have access also to the gradient and Hessian of the objective function respectively. We will use *black-box/zero order algorithm* to specify that the *oracle* used by the algorithm is black-box/zero order. *Black-box* is

a characteristic of the oracle. It means that the only information available for the method is the answers from the oracle.
The performance of a method $\mathcal{M}$ on $\mathcal{P}$ is the total amount of *computational effort* required by $\mathcal{M}$ to *solve* $\mathcal{P}$. We need to define more precisely *computational effort* and *solve*. These definitions can make a big difference on the performance of a given method. We will consider *computational effort* to be as the *complexity* of the problem $\mathcal{P}$ for the method $\mathcal{M}$: the number of calls to the oracle which is required to solve $\mathcal{P}$. The term *solve* is related to some *accuracy* that the method needs to accomplish for their approximation to be considered as a solution of the problem $\mathcal{P}$. One of the most natural ways to measure the accuracy is using the difference between the real optimum x^* and the approximated optimum output by the algorithm, $\hat{x}$ in the image space. That is $F(\hat{x}) - F(x^*) \leq \varepsilon$ (we say $\hat{x}$ an ε-solution).
The complexity and convergence rate are related. The convergence rate measures how fast the sequence of approximated points converges to the optimum. This can be measured also on the image space. If we have access to the convergence rate of the method, we can impose that $F(x_t) - F(x^*) \leq \varepsilon$ and obtain the t such that the solution at time t, x_t is an ε-solution of the problem. For more information, the reader can refer to [Nesterov, 2004].

2.3 Test Functions

Throughout this thesis we study the noisy counterpart of a deterministic function. We denote by F the deterministic function and f its noisy counterpart. Their definitions are as follows:

$$F : \mathcal{D} \to \mathbb{R}, \qquad x \mapsto F(x) \tag{2.2}$$

$$f : (\mathcal{D}, \Theta) \to \mathbb{R}, \qquad (x, \theta) \mapsto f(x, \omega(\theta)) = F(x) + \omega(\theta). \tag{2.3}$$

If $\mathcal{D} \in \mathbb{R}^d$ the problem is a *continuous problem* or if $\mathcal{D} \in \mathbb{Z}^d$ the problem is a *discrete problem*. We denote simply by $f(x, \omega(\theta)) = f(x, \omega)$. In order to lighten the notation, we will only use $f(x, \omega)$ when ω is relevant. Otherwise, we will use $f(x)$, always keeping in mind the dependence of ω, therefore the stochastic nature of f.

2.3.1 Sphere Function

A large part of the analysis presented in this thesis is based on the sphere function. We define here its basic form.

Definition 2.1. *Let $d \in \mathbb{N}$. Then the sphere function is defined as follows:*

$$\begin{aligned} F : \mathbb{R}^d &\to \mathbb{R} \\ x &\mapsto ||x - x^*||^2. \end{aligned}$$

with $||\cdot||$ the euclidean norm.

2.4 Modes of Convergence

For a deterministic sequence, there is only one definition of convergence:

Definition 2.2 (Convergence of a deterministic sequence)**.** *Let $(x_t)_{t\geq 1} \in \mathbb{R}^d$. Then $(x_t)_{t\geq 1}$ converges to x^* if for all $\varepsilon > 0, \exists T > 0$ such that for all $t \geq T$,*

$$||x_t - x^*|| \leq \varepsilon.$$

The convergence uses the notations: $\lim_{t\to\infty} x_t = x^*$ *or* $x_t \to x^*$ *among others.*

Whilst for random variables there are severals definitions not equivalent. In some cases one type of convergence is sufficient to have another type of convergence. For instance, convergence in probability (Definition 2.4) is a consequence of the almost sure convergence (Definition 2.3). Nonetheless, the reciprocal is not true.

Definition 2.3 (Almost sure convergence)**.** *The sequence $\mathcal{X}_t$ of random variables converges to a random variable $\mathcal{X}$ almost surely (a.s.) if*

$$\mathbb{P}(\lim_{t\to\infty} \mathcal{X}_t = \mathcal{X}) = 1.$$

Remark 2.1. *Almost sure convergence is equivalent to convergence with probability one.*

Definition 2.4 (Convergence in probability)**.** *The sequence $\mathcal{X}_t$ converges in probability to a random variable $\mathcal{X}$ if for all $\varepsilon > 0$*

$$\lim_{t\to\infty} \mathbb{P}(||\mathcal{X}_t - \mathcal{X}|| \geq \varepsilon) = 0.$$

2.5 Convergence Order of Deterministic Sequences

Let a deterministic sequence $(x_t)_{t\geq 1}$ converge to $x \in \mathbb{R}^d$, and assume for all $t, x_t \neq x$. Let

$$\lim_{t\to\infty} \frac{||x_{t+1} - x||}{||x_t - x||^q} = \mu, \qquad \text{with } q \geq 1, \mu \in (0, 1). \tag{2.4}$$

We define the *order of convergence of the sequence* $(x_t)_{t\geq 1}$ depending on q and μ in Definitions 2.5, 2.6 and 2.7. They correspond to the *super-linear, linear and sub-linear* convergence.

Definition 2.5 (Super-linear)**.** *The order of convergence is* super-linear *if any of the following conditions is satisfied:*

- $\mu = 0$,
- $q > 1$,
- $\mu > 0$. *In this case we say* convergence with order $q > 1$. *If $q = 2$, we say the convergence is* quadratic.

Definition 2.6 (Linear)**.** *The order of convergence is* linear *if* $q = 1$ *and* $\mu \in (0, 1)$*. In this case we obtain:*

$$\lim_{t\to\infty} \frac{||x_{t+1} - x||}{||x_t - x||} = \mu \ . \tag{2.5}$$

Definition 2.7 (Sub-linear)**.** *The order of convergence is* sub-linear *if* $q = 1$ *and* $\mu = 1$*. We can extend the definition to* convergence with degree $p > 0$ *if*

$$\lim_{t\to\infty} \frac{||x_{t+1} - x||}{||x_t - x||} = 1 - c_t||x_t - x||^{1/p} \ , \qquad \text{with } c_t \to c > 0 \ . \tag{2.6}$$

Note that if $p \to \infty$ *and* $c < 1$ *we get linear convergence and note that Equation 2.6 implies*

$$||x_t - x|| \sim \left(\frac{p}{c}\right)^p \frac{1}{t^p}$$

2.6 Convergence of Random Variables

Consider now $(x_t)_{t\geq 1}$ a sequence of random variables. The definitions in Equations 2.4 and 2.5 are no longer appropriate in general. We define a weaker version of linear convergence, analogous to the definition linear convergence in 2.5 (see [Auger and Hansen, 2011] for details).

Definition 2.8 (Log-linear convergence)**.** *The sequence* $(x_t)_{t\geq 1}$ *converges* log-linearly *almost surely if* $\exists c > 0$ *such that:*

$$\lim_{t\to\infty} \frac{1}{t} \log \frac{||x_t - x||}{||x_0 - x||} = c \quad a.s. \tag{2.7}$$

The sequence $(x_t)_{t\geq 1}$ *converges log-linearly in expectation or in mean if* $\exists c > 0$ *such that:*

$$\lim_{t\to\infty} \frac{1}{t} \mathbb{E}\left(\log \frac{||x_t - x||}{||x_0 - x||}\right) = c \ . \tag{2.8}$$

Following the same notation as in deterministic sequences, if the sequence $||x_t - x||$ converges to 0 following $1/t^p$ we say that the sequence *converges sub-linearly with degree p.*

2.6.1 Simple and Cumulative Regret

In this thesis we fix our attention to one parameter to determine the rate of convergence of an algorithm. First of all, allow us to make a difference between *search* and *recommendation* points. The *search points* are all the points explored by the algorithm where the objective function is evaluated. The *recommendation points* are the points the algorithm output as approximations to the optimum. The search and recommendation points can be the same, but note that the algorithm could produce recommendation points without having to evaluate them.
In this section we will define Simple and Cumulative Regret. The Simple Regret refers to the most natural way to measure the accuracy of an optimization method: measure the difference between the *recommendation* and the optimum. In other words, we only look at the points the algorithm outputs as approximations to the optimum. On the contrary, the Cumulative Regret, as

the name suggests, considers the cumulative *cost* of evaluating points. Therefore, in this case we consider *all* the points that are being evaluated throughout the optimization process. Therefore all the search points are considered. We denote by x_t the sequence of search points and $\hat{x}_t$ the sequence of recommendation points. Following the definition of sub-linear convergence, we obtain $||\hat{x}_t - x|| \sim \frac{1}{t^p}$. Or, in other words

$$\log(||\hat{x}_t - x||) \sim -p \log(t) \ .$$

Therefore, as $-p < 0$, then $||x_t - x||$ is converging to 0. And if we look at a plot using as x-axis the $\log(t)$ and the y-axis as $\log(||\hat{x}_t - x||)$ we obtain that the line on the plot should have *slope* $-p$. When $-p$ is close to 0, then the approximations approach the optimum slowly (if approaching at all). When $-p$ negative with large absolute value, that means that the approximations approach to the optimum fast.

We denote the *Simple Regret at iteration t* by SR_t and define it as follows, considering the recommendation points and their distance to the optimum x^* in the image space instead of the search space.

$$SR_t = \mathbb{E}_\omega(f(\hat{x}_t, \omega) - f(x^*, \omega)) = F(\hat{x}_t) - F(x^*) \ . \tag{2.9}$$

The last equality is obtained using the fact that the noise is unbiased. Note that we need to consider the expectation since the function f is stochastic. Since the convergence order in the cases studied in this thesis are usually *sub-linear*, to study the convergence rate we will observe the behaviour of Simple Regret SR_t with regards to t. That is, we will observe the *slope* on the log-log graph. We define then the *slope of the Simple Regret*[1] (denoted $s(SR)$) as follows

$$s(SR) = \lim_{t \to \infty} \frac{\log(SR_t)}{\log(t)} \ . \tag{2.10}$$

We will also consider another accuracy measure: the *Cumulative Regret* (denoted CR). The CR is used widely in the bandit community in order to measure the *expected gain (or loss)* when there are several options to take. It considers *all* the points that are evaluated, measuring therefore the cost of exploration.

$$CR_t = \mathbb{E}_\omega \left[\sum_{\tau=0}^{t} (f(x_\tau, \omega) - f(x^*, \omega)) \right] = \sum_{\tau=0}^{t} F(x_\tau) - F(x^*) \ . \tag{2.11}$$

To study the convergence rate we define the slope of the Cumulative Regret analogous to the Simple Regret

$$s(CR) = \lim_{t \to \infty} \frac{\log(CR_t)}{\log(t)} \ . \tag{2.12}$$

Note that it is equivalent to say $s(SR) = -1$ and $SR_t = O(t^{-1})$, with $O(\cdot)$ the usual Landau notation.

[1] The *slopes* are exactly the *convergence rates*.

2.7 Useful tools

2.7.1 Properties of Random Variables

Let $\mathcal{X} : \Omega \to \mathbb{R}$ be a random variable.

Lemma 2.1 (Linearity of Expectations)**.** *Let $\mathcal{X}, \mathcal{Y}$ be random variables, $\alpha, \beta \in \mathbb{R}$. Then,*

$$\mathbb{E}[\alpha\mathcal{X} + \beta\mathcal{Y}] = \alpha\mathbb{E}[\mathcal{X}] + \beta\mathbb{E}[\mathcal{Y}] \ .$$

Lemma 2.2 (Independence)**.** *Let $\mathcal{X}, \mathcal{Y}$ be independent random variables. Then*

$$\mathbb{E}[\mathcal{X}\mathcal{Y}] = \mathbb{E}[\mathcal{X}]\mathbb{E}[\mathcal{Y}] \ .$$

Lemma 2.3 (Properties of the Variance)**.** *Let $\mathcal{X}, \mathcal{Y}$ be random variables, $\alpha, \beta \in \mathbb{R}$. Then*

1. $Var[\alpha] = 0$.
2. $Var[\alpha + \beta\mathcal{X}] = \beta^2\, Var[\mathcal{X}]$.
3. $Var[\mathcal{X} + \mathcal{Y}] = Var[\mathcal{X}] + Var[\mathcal{Y}]$.
4. $Var[\mathcal{X}] = \mathbb{E}[\mathcal{X}^2] - (E[\mathcal{X}])^2$.

Lemma 2.4 (Transformation of Random Variables)**.** *Let $\mathcal{X}$ be an absolute continuous random variable with density $f_{\mathcal{X}}$. Let $\beta \neq 0$, $\alpha \in \mathbb{R}$. Then the random variable $\mathcal{Y} = \alpha + \beta\mathcal{X}$ is absolute continuous with density $f_{\mathcal{Y}}$ given by:*

$$f_{\mathcal{Y}}(y) = \frac{1}{|\beta|} f_{\mathcal{X}}\left(\frac{y-\alpha}{\beta}\right), \qquad y \in \mathbb{R}.$$

2.7.2 Inequalities

Lemma 2.5 (Markov's inequality)**.** *Let $\mathcal{X}$ be a non-negative random variable. Then for any real number $\varepsilon > 1$*

$$\mathbb{P}[\mathcal{X} \geq \varepsilon\mathbb{E}[\mathcal{X}]] \leq \frac{1}{\varepsilon} \ .$$

Lemma 2.6 (Chebyshev's inequality)**.** *Let $\mathcal{X}$ be a random variable with finite expectation μ. Then for any real number $k > 0$*

$$\mathbb{P}[|\mathcal{X} - \mu| \geq \varepsilon] \leq \frac{\mathbb{E}[|\mathcal{X} - \mu|]^2}{\varepsilon^2} \ .$$

Lemma 2.7 (Bertrand series)**.** *The Bertrand series is defined as $\sum_{i=2}^{n} \frac{1}{i \log i^{\beta}}$ and*

$$\lim_{n\to\infty} \sum_{i=2}^{n} \frac{1}{i \log i^{\beta}} < \infty \ , \qquad \text{for any } \beta > 1.$$

2.7.3 Bounds

Lemma 2.8. *Let $a, b \in \mathbb{R}_+$ such that $b \geq 1$. Then*

$$(1-a) \leq b(1 - a^{\frac{1}{b}}) \ .$$

Chapter 3

State of the Art

This chapter will present the key elements in the literature that inspired the work of this thesis. We present basically three sections: Deterministic Optimization, Stochastic Optimization and Random Search Heuristics. An important remark about the differences between the different types of algorithms presented here is the presence of "stochasticity" in either the problem or the algorithm. When we speak about Deterministic Optimization we will refer exclusively to problems and algorithms that are deterministic. When we refer to Stochastic Optimization we will refer to stochastic (or noisy) problems and deterministic or randomized algorithms. And when we speak about Randomized Search Heuristics, we will study methods that are randomized and solve deterministic and/or noisy problems.

Section 3.1 summarizes the results of classic optimization methods. We start with a general non-convex unconstrained optimization problem, where the functions are smooth and noise-free. The methods can be roughly classified into two categories: *Linesearch methods* and *Trust Region methods*. The Linesearch methods use a descent direction at which the value of the objective function is less than the current one and iteratively move along a descent direction. On the other hand, the Trust Region methods use a region where they approximate the function using a model.

In Section 3.2 we start by discussing the issues that arose when we have to optimize a noisy function. We explain some techniques used in the literature to deal with stochastic objective functions. And we present several algorithms, comparable to the linesearch methods of Section 3.1, that solve noisy functions.

Finally in Section 3.3 we present the Randomized Search Heuristics. These are optimization methods for global convergence that use a *population* of search points in order to obtain an improvement of their *fitness* (or objective function) after each iteration. These algorithms have an asset: they can deal with continuous or discrete functions, deterministic or noisy. The disadvantage is that the theoretical analysis is difficult.

In this thesis we work mostly with the continuous setup (Chapters 4, 5, 6, 7) but we also present a contribution in noisy discrete optimization problems. We will present the problem and the state of the art relevant for our contribution in Chapter 8.

3.1 Deterministic Optimization

In the field of Non-convex unconstrained optimization we consider the problem $\mathcal{P}$: $\min_{x \in \mathbb{R}^d} F(x)$. Where $F \in \mathscr{C}^1(\mathbb{R}^d)$ or $\mathscr{C}^2(\mathbb{R}^d)$. In this field, the literature is usually divided into *Linesearch methods* and *Trust Region methods*. We will give special emphasis to the Linesearch methods,

because we will present a contribution that uses two Linesearch methods in the context of noisy optimization (see Chapter 4).

3.1.1 Linesearch Methods

This section is based on the work of [Nesterov, 2004]. We present the results of classic algorithms for unconstrained optimization. This section presents only some important results and not to their proofs (for details the reader should refer to [Nesterov, 2004]). We will not do a profound analysis of each of the algorithms e.g. we do not state the exact conditions for convergence or analyze each proof of convergence. Instead, we will cover their convergence behaviour in a general way, observing their dependence on the iteration and other relevant parameters. From the *Gradient method*, with only smoothness required, until the treatment of strongly convex functions, we will see how the algorithms are conceived to solve specific "well behaved" functions. The properties from the objective functions, inherited by the algorithms, make them specialized and efficient algorithms.

3.1.1.1 The Gradient method

The Gradient method is a simple scheme used in unconstrained optimization. It takes advantage of the fact that the antigradient is a direction of steepest descent of a differentiable function. The scheme is in Algorithm 3.1.

Algorithm 3.1 Gradient method

Initialize Choose initial solution $x_0 \in \mathcal{D}$

1: **while** termination criterion not reach **do**
2: $\quad x_{t+1} = x_t - \sigma_t F'(x_t)$
3: $\quad t = t + 1$
4: **end while**

In Algorithm 3.1, $F'(x)$ represents the gradient of F and σ_t is called the *step size* at iteration t. There are different ways to choose this step size depending on the context. The step size can be constant, depending on t or following the Goldstein-Armijo rule.
Let us assume that $F \in \mathscr{C}_M^{2,2}(\mathbb{R}^d)$, that $\exists x^*$ local minimum of F at which the Hessian is positive definite. Also we assume the Hessian is bounded at x^*: there exists $0 < l < L < \infty$ such that $l\mathbb{I} \leq F''(x^*) \leq L\mathbb{I}$. Finally, the initial solution x_0 is close enough to the optimum. Then we know the local convergence rate of the method:

Theorem 3.1. *Let F be a function that satisfies the previous assumptions and let x_0 such that $r_0 = ||x_0 - x^*|| \leq \bar{r} = \frac{2l}{M}$ Then the Gradient method with optimal step size $\sigma_t = \frac{2}{L+l}$ converges with rate:*

$$||x_t - x^*|| \leq \frac{\bar{r} r_0}{\bar{r} - r_0} \left(1 - \frac{l}{l+L}\right)^t. \tag{3.1}$$

The former convergence rate is *linear*. Recall that a linear convergence rate is given in terms of an exponential function of the iteration counter t. We can deduce directly that the corresponding complexity bound (the number of evaluations before achieving ε accuracy) is $\frac{1}{q}\left(\log c + \log\frac{1}{\varepsilon}\right)$ where $c = \frac{\bar{r}r_0}{\bar{r}-r_0}$ and $q = \frac{l}{l+L}$.

3.1.1.2 Newton method

Initially proposed to solve a root problem, the Newton method holds some differences with the Gradient method. The iterative process uses second order information as opposite to the gradient method that uses only first order information. The iteration is then $x_{t+1} = x_t - [F''(x_t)]^{-1}F'(x_t)$. The iterative step can be deduced either from the extension of a root problem to the solution of a system of nonlinear equations or from the use of the idea of a quadratic approximation. The Newton method presents two serious disadvantages: first if $[F''(x_t)]$ is degenerate then the process breaks and second the possible divergence. We present in algorithm (3.2) the Newton method.

Algorithm 3.2 Newton method

Initialize Choose initial solution $x_0 \in \mathcal{D}$

1: **while** termination criterion not reach **do**
2: $\quad x_{t+1} = x_t - [F''(x_t)]^{-1}F'(x_t)$
3: $\quad t = t + 1$
4: **end while**

We can use similar strategies as the Gradient method to choose the step size. For the study of local convergence we assume that $F \in \mathscr{C}_M^{2,2}(\mathbb{R}^d)$, that $\exists x^*$ with positive definite Hessian: $F''(x^*) \leq l\mathbb{I}, l > 0$ and that the initial solution x_0 is close enough to x^*.

Theorem 3.2. *Let F a function that satisfies the previous assumptions and let x_0 such that $r_0 = ||x_0 - x^*|| \leq \bar{r} = \frac{2l}{3M}$. Then the Newton method converges with rate:*

$$||x_{t+1} - x^*|| \leq \frac{M||x_t - x^*||^2}{2(l - M||x_t - x^*||)}. \tag{3.2}$$

The former convergence rate is *quadratic*. Recall that a quadratic convergence rate is given in terms of a double exponential function of the iteration counter t. We can see then that the Newton method is much faster than the gradient method, with a smaller convergence region similar, but smaller.

3.1.1.3 Quasi-Newton and Conjugate Gradient

The Gradient method and the Newton method have different convergence rates due to the direction used to minimize each iteration. The Gradient method uses the antigradient, measured on euclidean norm. On the other hand, the Newton method uses the Hessian. A deeper analysis on the Hessian yields a more significant result. The Newton method uses the gradient computed with respect to the metric defined by $F''(x)$.

The Quasi-Newton (or variable metric) methods arise with ways to approximate the Hessian instead of computing it. They compute iteratively some $H_t \to H''(x^*)$ to use in the iterative computation of the points x_t. For quadratic functions the method usually finishes in d iterations. In general there is a local convergence in the neighborhood of an strict minimum with a superlinear rate of convergence. The main disadvantage of Quasi-Newton methods its the necessity to storage a matrix of size $d \times d$ for each iteration. This is the motivation for the development of conjugate gradient methods.
Conjugate gradient methods are initially proposed to solve quadratic functions and in this context they use the structure of a quadratic function to narrow the minimization process. This leads to the computation of some specific directions *conjugate* with respect to the matrix that defines the objective function, as many direction as the dimension of the problem. The method can be extended to optimize any non-linear function, and hopefully the method keeps all its good properties when we are close to the optimum, where the function is actually close to a quadratic one.
If the function to optimize is in fact quadratic, then the method finishes in d iterations. For any nonlinear function it may happen that after d iterations the direction is not correct. In that case we can implement some *restart* strategy that ensures global convergence of the method. The global convergence is not ensured for general conjugate gradient methods. The method also exhibits local quadratic convergence, slower than the convergence of the Quasi-Newton methods. However, the advantage of having "light" iterations allows it to continue on the shortlist for the choice of an algorithm.

3.1.1.4 Smooth Convex Programming

In the previous section we talked about optimizing functions assuming very weak assumptions about them, only $F \in \mathscr{C}^1(\mathbb{R}^d)$ or $\mathscr{C}^2(\mathbb{R}^d)$. In that context the gradient method can only ensure convergence to a stationary point. We would like to have a family of functions where the first order optimality is sufficient for the solution to be global. Also we would like to maintain some generality on the family, so that the family is closed under addition and multiplication by a positive constant. Finally we want to include the linear functions. We therefore introduce the family of differentiable *convex* functions.

Definition 3.1 (Convex)**.** *A continuous differentiable function $F(x)$ is called* convex *if for any $x, y \in \mathbb{R}^d$ we have*

$$F(y) \geq F(x) + F'(x) \cdot (y - x).$$

This class also has the advantage to have many equivalent definitions ([Nesterov, 2004]). We will denote the set of convex functions by $\mathscr{F}_L^{k,l}(Q)$ with the index having the same meaning as for families of continuous functions. Some example of members of $\mathscr{F}_L^{k,l}(Q)$ are the linear functions, the quadratic functions, the exponential, $F(x) = |x|^p$, among others.

Complexity Lower Bound for $\mathscr{F}_L^{1,1}(\mathbb{R}^d)$

Theorem 3.3. *For any t, $1 \leq t \leq \frac{1}{2}(d-1)$ and any $x_0 \in \mathbb{R}^d$, there exists $F \in \mathscr{F}_L^{1,1}(\mathbb{R}^d)$ such that for any first order method $\mathcal{M}$ satisfying milder conditions we have*

$$f(x_t) - f^* \geq \frac{L||x_0 - x^*||}{8(k+1)^2}, \tag{3.3}$$

$$||x_t - x^*||^2 \geq \beta||x_0 - x^*||, \tag{3.4}$$

with β arbitrarily close to 1.

We find then that the lower bound for Simple Regret is good. For instance by iteration 100 we already have a reduction of the difference between x_t and x^* of 10^4. However, the convergence of the sequence of search points can be arbitrarily slow.

Strongly Convex Functions We introduce the class of Strongly Convex Functions, where we will find a linear rate of convergence of the gradient method, but now in a global way. The strongly convex functions families is denoted by $\mathscr{S}_L^{1,1}(\mathbb{R}^d)$, where the index have the same meaning as for families of continuous functions. For instance the function $F(x) = \frac{1}{2}||x||^2$ is a strongly convex function.

Definition 3.2 (Strongly Convex)**.** *F continuously differentiable is called* strongly convex *if there exists a constant $\mu > 0$ such that for any $x, y \in \mathbb{R}^d$ we have:*

$$F(y) \geq F(x) + F'(x) \cdot (y - x) + \frac{1}{2}\mu||y - x||^2.$$

Complexity Lower Bound for $\mathscr{S}_L^{1,1}(\mathbb{R}^d)$

Theorem 3.4. *For any $\mu > 0$, $Q_f > 1$ and any $x_0 \in \mathbb{R}^d$, there exists $F \in \mathscr{S}_L^{1,1}(\mathbb{R}^d)$ such that for any first order method $\mathcal{M}$ satisfying mild conditions we have:*

$$f(x_t) - f^* \geq \left(\frac{\sqrt{Q_f - 1}}{\sqrt{Q_f + 1}}\right)^{2k} ||x_0 - x^*||^2, \tag{3.5}$$

$$||x_t - x^*||^2 \geq \frac{\mu}{2}\left(\frac{\sqrt{Q_f - 1}}{\sqrt{Q_f + 1}}\right)^{2k} ||x_0 - x^*||^2, \tag{3.6}$$

with β arbitrarily close to 1.

Gradient method performance on $\mathscr{F}_L^{1,1}(\mathbb{R}^d)$ and $\mathscr{S}_L^{1,1}(\mathbb{R}^d)$

Theorem 3.5. *If $F \in \mathscr{F}_L^{1,1}(\mathbb{R}^d)$ and $h = \frac{1}{L}$, then*

$$f(x_t) - f^* \geq \frac{2L||x_0 - x^*||}{k+4}. \tag{3.7}$$

If $F \in \mathscr{S}_L^{1,1}(\mathbb{R}^d)$ and $h = \frac{2}{\mu+L}$, then

$$f(x_t) - f^* \geq \left(\frac{L-\mu}{L+\mu}\right)^{2k} ||x_0 - x^*||, \tag{3.8}$$

$$||x_t - x^*|| \geq \frac{L}{2}\left(\frac{L-\mu}{L+\mu}\right)^{k} ||x_0 - x^*||. \tag{3.9}$$

Comparing these convergence rates with the lower bounds in Section 3.1.1.1 and 3.1.1.2 we can see that the lower bounds are not met by the Gradient method. Note that the analysis of the convergence rate on the gradient method relies on a relaxation sequence $f(x_{t+1}) \leq f(x_t)$. Nonetheless, the construction of optimal methods in convex optimization never relies on the use of a relaxation sequence. This is because it might be too expensive to ensure a relaxation sequence and because the methods are derived from global topological properties of convex functions, whilst the relaxation sequence exhibit a "microscopic" relationship.

3.1.2 Trust region Methods

As we saw in the previous section, non-linear optimization problems are solved numerically by iterative algorithms. At each iteration there is a *current* recommendation and a new point is generated using some guidelines.

Trust Region methods are relatively new compared to the line search algorithms[1]. They use recommendations points and an approximate model constructed close to the point. Then they can use this model to generate the new recommendation. The name "Trust Region" refers to the criterion: the model is trusted *only* in a region around the current recommendation. The models can be linear or quadratic (or higher orders), then it makes sense to create a region where the model will approximate well the function. The region evolves at each iteration: if the model is a good fit for the problem, then the region is enlarged. On the contrary, if the difference between the model and the real problem is big, then the region is shrunk.

The first works on Trust Region methods were done by Powell [Powell, 1970a; Powell, 1970b]. For a recent survey the reader can explore [Conn *et al.*, 2000]. The trust region methods have good convergence results. They can be faster and more robust than Linesearch algorithms ([Yuan, 2000]).

In this thesis we will not study this type of algorithms. Our main motivation is keeping the information to the minimum and study zero order methods. Even when we will analyze the noisy version of the Newton method (Section 3.1.1.2), we assume to have function evaluations used to *estimate* the Hessian, and not the values of the Hessian. Nonetheless, the analysis of Trust Region methods in the context of noisy optimization is an attractive track, as proved by the work of [Elster and Neumaier, 1997] and the recent work of [Cartis and Scheinberg, 2015; Chen *et al.*, 2015].

3.2 Stochastic Optimization

This section is based on the article of [Spall, 2003]. The results presented in this section will typically not be proven. We choose to not write the details of the proofs to observe the general behaviour of the algorithms in terms of convergence. To see the details the reader can refer to the references.

[1] This section is based on the Review from [Yuan, 2000]

We address in this section stochastic optimization algorithms of zero order, therefore they only use function evaluations. Therefore we exclude the algorithms that have access to gradients or higher order information.
We consider the continuous minimization problem detailed in Chapter 2. It is usually assumed that F is smooth to some degree. The question of attaining a global or local optimum is an important issue. Here we considerer a slightly different concept than the one used on the previous section. In Section 3.1 we saw how classical algorithms obtain *local convergence*, meaning that they converge to the optimum if the initial point is inside some region. Here we speak about *local convergence* when we can ensure that the algorithms will reach a local minimum, regardless of the initial point.
In general in stochastic optimization the goal is to ensure that the algorithm will reach a local optimum in a finite number of iterations or function evaluations. The scope of the certainty of the convergence is limited since the algorithms are very general and they use (on purpose) as little information as possible about the objective function. So it is easy to design special functions in order to prove that the algorithm can fail. Therefore we will present here algorithms that will improve the solutions, but sometimes they do not have the theoretical analysis that ensures global or local convergence.
When we speak about stochastic optimization we refer to two sources of noise in the optimization process.

- Noise in the function evaluation: usually as additive noise. Therefore the function evaluations have the form $f(x, \omega) = F(x) + \omega$ as in Section 2.3. The use of noisy function evaluation happens each time there is a physical measurement or some computer simulation process involved on the optimization.

- Noise in the algorithm: it modifies the search distribution through random choices. An example is the *mutation process* in the Evolution Strategies. The aim of introducing artificially noise in the algorithm is to force the diversification of candidate solutions. It is therefore a mean to speed up the convergence. It allows the algorithm to be more robust to model errors. Also, thanks to this random choices there is room for the exploration of the search space onto areas that were not explored in the past.

In contrast, the classical deterministic optimization algorithms (Section 3.1.1) assume that there is perfect information of the objective function, whether it is the objective function values, gradients or higher order. And this information is used to compute the search direction of the algorithm also in a deterministic way.

3.2.1 Analysis of Stochastic Optimization Algorithms

There are several issues common to the analysis of all stochastic optimization algorithms. First, there are fundamental limits in the optimization process when we have access to noisy function evaluations. Possibly the most important is that the statistical standard error can only be improved

by using a significant number of function evaluations. The classical result in statistics in the case of independent noise says that the error decreases at rate $1/\sqrt{n}$, where n is the number of function evaluations.
Second, the volume of the search space grows rapidly with the dimension. This calls for the use of some structure in the objective function to narrow the search.
Third, in any optimization process, whether it is stochastic or not, it is very difficult to design automated methods to stop the algorithm at some point. Without some previous knowledge of the function, in practice it is impossible to be sure about the quality of the solution found so far. Even more, in real applications the environment plays a key role and it might affect drastically from one moment to another. So even if we run an optimization method for some time and chose to stop and output some solution, we cannot know how this solution will perform when some changes are introduced in the environment.
Forth and last, the theorems of *No Free Lunch* (NFL) offer a formal context for an intuitive result: one algorithm cannot be effective on every problem. There is a trade-off between the performance of the algorithm and its robustness. The consequences of the NFL theorems are valid both in noisy and noise-free situations. The theory is established on discrete setting, but given that in practice all optimization problems are solved using computers, there is a translation between the discrete result to the optimization of continuous functions.

3.2.2 Techniques to handle noise

The problem of *misranking* appear in the comparison-based algorithms that use the *comparison* of the fitness values of the population to select the best points to approximate the optimum. In other words, if we consider individuals x_1 and x_2 and an additive noise model, then due to the noise perturbation we might obtain $f(x_1) > f(x_2)$ or equivalently in our model $F(x_1)+\omega_1 > F(x_2)+\omega_2$. Whereas actually the real ordering between individuals is the opposite, i.e. $F(x_1) < F(x_2)$. The following section present three techniques that help to reduce the effect of the noise in noisy optimization algorithms.
The following sections mention some techniques used in the literature to avoid problems of misranking. But not only that. The techniques are basically a way to obtain a better estimation of the real function values associated with a specific point x.

3.2.2.1 Reevaluation or Resampling

A simple way to handle the noise of the objective function values is the *reevaluation or resampling* of the search points[2]. The idea is to query the noisy objective function several times at each point and use a statistic to estimate the real function value. If the noise is unbiased, the use of the *mean* of the sample of objective values at a fix search point is a good choice that reduces the variance of the noise. For example, the Fabian algorithm [Fabian, 1967] is a Linesearch method for stochastic

[2]In this thesis we will use the term "reevaluation" to emphasize that we query several times the *evaluations* of specific points. The reader may find the term *resampling* in some of the references and in the literature.

optimization that uses reevaluation to estimate a gradient.
In this thesis we will study extensively this technique. In Chapter 4 we will estimate gradients and Hessians using the reevaluation of search points. In Chapter 5 we study the convergence of some specific Randomized Search Heuristics to optimize noisy functions. In Chapter 8 we study the runtime of algorithms solving noisy discrete functions, using reevaluation to cope with the noise.

3.2.2.2 Surrogate Models

Another way to handle the noise is to *learn* the objective function. This means, the algorithm samples several points where the model is not precise enough and then assume the real optimum is close to the optimum of the learnt model of the objective function. Such algorithms can be found in [Jones *et al.*, 1998; Leary *et al.*, 2004; Caballero and Grossmann, 2008; Villemonteix *et al.*, 2009]. If the reader is interested, note that the work for surrogate models is clearly influenced by their performance on real problems. In other words, most of the work in the literature refers to the study of specific cases and the generalization of the conclusions is not yet fully studied.
Note that we can also find surrogate models and evolutionary search combined [Ong *et al.*, 2003; Coulom *et al.*, 2011; Coulom, 2012].

3.2.2.3 Increase population

Another method used to reduce the influence of the noise is the increase of the population in Evolutionary Algorithms. One example is the work of [Arnold and Beyer, 2001].

3.2.3 Finite Differences

The cornerstone in the Stochastic Optimization domain is the appearance of the algorithms of Robbins-Monro [Robbins and Monro, 1951]. They introduce a general algorithm that finds the solution of a root problem using noisy function evaluations. The root problem can be viewed as the search for the solution of the Equation 3.10.

$$g(x) = \nabla F(x) = 0, \tag{3.10}$$

where F represents the deterministic underlying function that we are trying to optimize. We assume as usual that we do not have access to the real value of F but to a perturbed function value f as defined in Section 2.3. When there is direct access to the noisy evaluations of the *gradient* of F, then there exists a whole series of algorithms gathered by the name *stochastic gradient algorithms* that can solve 3.10. We will focus here on algorithms of zero order, therefore having access only to the function evaluations (with noise) and not the gradients.

If we compare with the classic gradient descent algorithm, described in Section 3.1.1.1, we can note that we no longer have access to $F'(x)$. Therefore the Finite Differences methods is the way to estimate the gradient, represented by $\hat{F}'(x)$. The update is therefore defined in Equation 3.11.

$$x_{t+1} = x_t - \alpha_t \hat{F}'(x), \tag{3.11}$$

where $\alpha_t > 0$. And therefore $\hat{F}'(x)$ is the estimation by finite differences using the noisy function evaluations available. The one sided Finite Differences was proposed by [Blum, 1954] in the context of Stochastic Optimization and uses $f(x)$ and $f(x + ce_i)$ where $(e_i)_{j=1}^d$ represents the canonical basis of $\mathbb{R}^d$,and $c > 0$. The two sided Finite Differences was proposed notably by [Fabian, 1967], Algorithm 3.3, and it uses $f(x \pm ce_i)$.

Algorithm 3.3 Fabian Algorithm. e_i is the i^{th} vector of the standard orthogonal basis of $\mathbb{R}^d$ and $e_{1,s/2}$ is the 1^{st} vector of the standard orthogonal basis of $\mathbb{R}^{s/2}$. v_i is the i^{th} coordinate of vector v. $(\hat{x}_i)$ is the i^{th} coordinate of intermediate points $(\hat{x})$. $(x^{(i,j)+})$ and $(x^{(i,j)-})$ are the search points and $\tilde{x}$ is the recommendation. Here, the index t is the number of *iterations*.

Input: An even integer $s > 0$. Parameters a, α, c, γ.
Initialization:
$u_i \leftarrow \frac{1}{i}$, $\forall\, i \in \{1, \dots, s/2\}$
Matrix $U \leftarrow \left(\|u_j^{2i-1}\|\right)_{1 \le i,j \le s/2}$
Vector $v \leftarrow \frac{1}{2}U^{-1}e_{1,s/2}$
$\tilde{x} \leftarrow x \in [0,1]^d$ uniformly at random
$n \leftarrow 1$
while not terminated **do**
    $a_t \leftarrow \frac{a}{t^\alpha}$, $c_t \leftarrow \frac{c}{t^\gamma}$
    $\forall j \in \{1, \dots, s/2\}$, $\forall i \in \{1, \dots, d\}$
    Evaluate:

$$x^{(i,j)+} \leftarrow \tilde{x} + c_t u_j e_i \qquad x^{(i,j)-} \leftarrow \tilde{x} - c_t u_j e_i$$

    $\hat{x}_i \leftarrow \frac{1}{c_t} \sum_{j=1}^{s/2} v_j \left(f(x^{(i,j)+}) - f(x^{(i,j)-})\right)$
    Recommend: $\tilde{x} \leftarrow \tilde{x} - a_t \hat{x}$
    $t \leftarrow t + 1$
end while
return $\tilde{x}$

In fact, in [Fabian, 1967] we observe an estimation of the gradient with finite differences and also repetition of the sample of the function value. An estimation of the gradient at x_t has the form:

$$\hat{F}'(x_t) = \begin{bmatrix} \dfrac{f(x_t + c_t e_1) - f(x_t - c_t e_1)}{2c_t} \\ \vdots \\ \dfrac{f(x_t + c_t e_d) - f(x_t - c_t e_d)}{2c_t} \end{bmatrix}. \tag{3.12}$$

The convergence analysis for algorithms using Finite Differences is rather similar to the one made for the pioneer algorithm of Robbins-Monro. However, there is one key difference. Robbins-Monro uses the fact that the gradient estimator is unbiased. Meanwhile the two finite differences

estimation methods presented here are biased. Therefore the finite difference method can be biased. Nonetheless the conditions imposed on α_t and c_t are fundamental:

- $\alpha_t > 0$ and $c_t > 0$.
- $\alpha_t \to 0$ and $c_t \to 0$.
- $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \frac{\alpha_t^2}{c_t^2} < \infty$.

The work in [Fabian, 1967] proves that the convergence rate for the mean square error $\mathbb{E}(\hat{x}_t - x^*)^2$ is $O(t^{-1})$ for functions that are smooth enough.
Note that in the former finite differences methods it is necessary to have at least $2d$ function evaluations in order to estimate the gradient. The work on [Spall, 2000] is also inspired from the use of Finite Differences but it reduces the amount of function evaluations to two per iteration to estimate the gradient, independent of the dimension d. The fact of having less function evaluations per iteration is only a significant advantage if the convergence rate does not deviate in such a way that it cancels the effect. In [Spall, 2000] we observe the proof under reasonable conditions the statistical accuracy for the mean square error is the same for both Finite Differences and Spall's variation. But the Spall variation uses only $1/d$ of the function evaluations for the classic Finite Differences methods in a given iteration.

3.2.3.1 Shamir Algorithm

In this thesis we will use a Linesearch algorithm for noisy optimization that obtains the direction from a one point estimation of the gradient. It will be used in this thesis mostly as an "optimal" algorithm to contrast empirical results. It is presented in [Shamir, 2013] as an algorithm for noisy quadratic functions. This algorithm attains a "fast" rate for the Simple Regret, and it is sharp. In this case $F(x)$ is a quadratic function and the observed noisy function evaluations are $f(x, \omega) = F(x) + \omega$ with noise ω_x distributes as a zero mean random variable with $\mathbb{E}(\omega_x^2) < \max\{1, ||x||\}$. The quadratic form F is defined in Equation 3.13.

$$\begin{aligned} F : \mathcal{D} &\to \mathbb{R} \\ x &\mapsto x^t A x + b^t x + c, \end{aligned} \tag{3.13}$$

where $\mathcal{D} \subset \mathbb{R}^d$, A is positive-definite, with spectral norm at most 1, $||b|| \leq 1$, and $|c| \leq 1$. These functions are both strongly convex and smooth.

Algorithm 3.4 Shamir's algorithm

Require: $\lambda > 0$; $\delta \in (0, 1]$

Initialize $x_1 = 0$

1: **for** $t = 1, \ldots, T$ **do**
2: Pick $r \in \{-1, +1\}^d$ uniformly at random
3: Query $v = f(x_t + \frac{\varepsilon}{\sqrt{d}} r)$
4: Let $\tilde{g} = \frac{\sqrt{d}v}{\varepsilon} r$
5: Let $x_{t+1} = \Pi_{\bar{\mathcal{D}}} \left(x_t - \frac{1}{\lambda t} \tilde{g} \right)$
6: **end for**
7: **return** $\hat{x}_T = \frac{2}{T} \sum_{t=T/2}^{T} x_t$

3.2.3.2 Convergence results of Shamir's algorithm

The work in [Shamir, 2013] exhibits an upper bound for Simple Regret of $\mathcal{O}(d^2/T)$ assuming that:

Assumption 1. *At least one of the following holds for some fixed $\varepsilon \in (0, 1]$:*

- *x^* belongs to the domain $\mathcal{D}$, at distance at least ε from the boundary of $\mathcal{D}$.*
- *The query points can be any point over $\mathcal{D}$ and any point at distance at most ε from $\mathcal{D}$.*

Then, the Theorem 3.6 presents the upper bound for the convergence.

Theorem 3.6. *Let F be a λ-strongly convex function as in (3.13), and suppose the optimum x^* has norm at most B. Then, under Assumption 1, the point x_T returned by Algorithm 3.4 satisfies*

$$\mathbb{E}(f(\hat{x}_T) - f(x^*)) \leq \frac{4(4 + 5\log(2))(B + 1)^4}{\lambda \varepsilon^2} \frac{d^2}{T}. \tag{3.14}$$

3.2.3.3 Convergence for Cumulative Regret

The work of [Shamir, 2013] also proves a lower bound for the Cumulative Regret (definition 2.11). This lower bounded is expressed by the following theorem.

Theorem 3.7. *Let $T > 0$. Then for any (possibly randomized) querying strategy, there exists a quadratic function of the form $f(x, \omega) = \frac{1}{2}||x||^2 - (x^* \cdot x) + \omega$, minimized at x^* such that $||x^*|| \leq 1/2$, such that*

$$\mathbb{E}\left[\frac{1}{T} \sum_{t=1}^{T} f(x_t) - f(x^*) \right] \geq 0.02 \min\left\{ 1, \sqrt{\frac{d^2}{T}} \right\}. \tag{3.15}$$

This lower bound is sharp up to constants, thanks to the result on [Agarwal *et al.*, 2010] shown for strongly-convex and smooth functions. The result on Theorem 3.7 implies that is not possible to obtain a better regret for more general problems that include forms of the type described in the theorem.

3.3 Randomized Search Heuristics

We have seen in Sections 3.1.1 and 3.1.2 that the construction of the methods heavily relies on the class of functions the algorithms solve. Especially to ensure global convergence, we design the algorithms hand in hand with the problem (e.g. see Section 3.1.1.4). We present here the Randomized Search Heuristics that are general heuristics/algorithms used to find global optimums. Instead of being designed specifically to solve a problem, they are population-based methods and use very simple mechanisms to ensure the improvement of the population in each iteration of the algorithms. We will describe in more detail the Evolutionary Strategies, part of the Evolutionary Algorithms. Evolutionary Algorithms together with other methods such as Simulated Annealing, Differential Evolution and Particle Swarm Optimization are considered as *global optimization algorithms*. For a complete revision of the latest results for Randomized Search Heuristics, consult [Auger and Hansen, 2011].
Randomized Search Heuristics are very easy to implement and hard to analyze. Their properties are not easy to grasp and there has been the development of tools to analyze the algorithm. The tools are usually simple and the proofs require ingenious ways to overcome the lack of substantial properties, as opposed to the analysis of Linesearch and trust region methods, where the structure of the functions is used extensively.

Randomized Search Heuristics are generally zero order methods that explore the space by sampling according a continuous search distribution. The simplest one is called Random Search proposed by [Brooks, 1958] that uses a fixed distribution to sample the exploration points and recommends the best solution so far.

3.3.1 Random Search

Random Search methods have the advantage of being easy to implement zero order methods. And especially their broad applicability makes them attractive. They can be used to optimize discrete, continuous and even hybrid functions [Gentle *et al.*, 2012]. The Random Search algorithm presented on Algorithm 3.5 (also called Blind Random Search) is the most basic Randomized Search Heuristic where the current solution does not depend on the previous solutions.
If the search space $\mathcal{D}$ corresponds to a hypercube, then the search distribution is usually the uniform distribution on $\mathcal{D}$. When the search spaces are more complicated we can use other techniques, such as rejection methods or some Monte Carlo generation of search points. For example we could use a uniform distribution on an hypercube containing $\mathcal{D}$ and checking if the search point belongs to $\mathcal{D}$. If not, we simply generate new search points until one is in fact a feasible search point.

Algorithm 3.5 Random Search.

Initialize: Candidate solution $\hat{x}$ randomly drawn in $\mathcal{D}$
bestFitness $\leftarrow f(\hat{x})$
Initialize: $t \leftarrow 1$
while not terminate **do**
    Randomly draw x in $\mathcal{D}$.
    fitness $\leftarrow f(x)$
    if fitness $<$ bestFitness **then**
        $\hat{x} \leftarrow x$
        bestFitness $\leftarrow$ fitness
    end if
    $t \leftarrow t + 1$
end while
return $\hat{x}$

The work on [Spall, 2003] ensures almost sure convergence under general conditions for the Random Search. While the convergence rate is reasonable when the dimension is low, Random Search is proved to be a very slow algorithm even for a moderate dimension of $\mathcal{D}$. This is a direct consequence of the exponential growth of the volume of the search space as d increases.

3.3.2 Adaptive Search Algorithms

The random search presented in the previous section can use a more sophisticated search distribution. Instead of sampling uniformly in all the search space at each iteration, we can design a sampling strategy that gives more importance to the search points near the current solution. This algorithm is called Localized Random Search (or in the Evolutionary Algorithm community: $(1 + 1)$ -ES where ES stands for Evolutionary Strategy). Where *localized* refers to the search distribution, not to the quality of the minimum the algorithm finds (global or local).
For continuous problems the modification involves an unbiased multivariate distribution and a variance such that for each component there is consistency with the magnitude of the vectors on $\mathcal{D}$. Therefore it assigns roughly the same importance to each of the components on $x \in \mathcal{D}$.
The concept can be expanded into algorithms that *adapt* the search distribution according to the problem. Therefore the name Adaptive Search Algorithms. It gives more importance to the search points *close* to the current recommendation, assuming that good solutions are usually close better solutions and thus closer to the optimum. We present in algorithm 3.6 the simplest adaptive search Algorithm, the $(1 + 1)$-ES, introduced by [Rechenberg, 1973] and [Schwefel, 1981].

Algorithm 3.6 $(1+1)$-ES algorithm

Initialize Candidate solution $\hat{x}$, $t = 1$

1: bestFitness $\leftarrow f(\hat{x})$
2: **while** termination criterion not reach **do**
3: Sample offspring $x = \hat{x} + \sigma_n \Phi$
4: fitness $\leftarrow f(x)$
5: **if** fitness $<$ bestFitness **then**
6: $\hat{x} \leftarrow y$
7: bestFitness $\leftarrow$ fitness
8: **end if**
9: $t \leftarrow t + 1$
10: **end while**

Note that the only difference with Random Search is the search distribution. In this case it depends on the current solution $\hat{x}$ and σ_n, which we will call *step-size*, and a distribution Φ. Typically Φ is a multivariate random distribution with zero mean, often a Gaussian. The question on how to adapt the dispersion of the search distribution has been addressed by many researchers. An important result is the 1/5-th sucess rule, presented by [Schumer and Steiglitz, 1968], that maintains a constant probability of success of around 1/5. The probability of success is the probability that the new search point has a smaller function value than the current recommendation. Another important result is the one presented by [Hansen and Ostermeier, 2001] that proposes to adapt the whole covariance matrix and not only the step-size of the search distribution. The algorithms is called CMA-ES.

Evolutionary Algorithms (EA) are known for using a whole *population* of search points in each iteration. We present here the algorithm (μ, λ)-ES (in Algorithm 3.7). At each iteration the algorithm generates λ search points (called *offsprings*) from the current recommendation point, using some search distribution. Then, it selects the best μ offsprings to compute the recommendation point, which will be the parent of the next iteration (or *generation*). Note that there is another difference with regards to the $(1+1)$-ES: the parent is excluded from the selection for the next generation. The algorithm that present this feature are called *non-elitist*. While the $(1+1)$-ES does take into account the parent on the selection for the best points. This fact is explicit by using "+" or "," in the definition of the algorithm.

Algorithm 3.7 (μ, λ)-ES algorithm

Initialize Parent x_1, $t = 1$

1: **while** termination criterion not reach **do**
2: **for** $j = 1, \ldots, \lambda$ **do**
3: Sample λ offsprings $P_\lambda^t = \{x_{t,1}, \ldots, x_{t,\lambda}\}$ by $x_{t,j} = x_t + \sigma_t \mathcal{N}$ ▷ Mutation
4: Evaluate fitness $y_{t,j} = f(x_{t,j}, \omega)$ ▷ Evaluation
5: **end for**
6: Select from P_λ^t the μ best search points with largest fitness values ▷ Selection
7: Update x_t and σ_t using μ best search points and its fitness evaluations
8: $t = t + 1$
9: **end while**

3.3.2.1 Invariance to Order Preserving Transformation

One important feature of ES is that they are invariant to monotonically increasing transformations. This is explained because of the nature of the selection process of the best offsprings in each generation. The use of the rank based on the function values of the offsprings allows that the algorithm optimizes function F or $g \circ F$ is exactly the same for the ES when g is a monotonically increasing function.

3.3.3 Convergence Results

The object of study is not only the convergence of the algorithms but also the convergence rate in order to have interesting results in practice. A popular accuracy measure in the EA community is the concept of *hitting time* τ_ε, defined as:

$$\tau_\varepsilon = \inf\{t : x_t \in B(x^*, \varepsilon)\}, \tag{3.16}$$

where $B(x^*, \varepsilon)$ is the ball centered over x^* and with radius ε. An equivalent definition can be used with the points on the image of F, so that we measure the accuracy in terms of F.

Random Search and (1+1)-ES converge with probability one to the global optimum in functions satisfying some general assumptions.The main assumption is that the global optimum is reachable by the search distribution with probability strictly positive in the neighbourhood of the optimum. In 1965 [Matyas, 1965] proves for the Localized Random Search that it converges with probability one for F continuous. The convergence rate for the hitting time of (1+1)-ES is $\Theta(1/\varepsilon^d)$, considering a slightly modified version of the hitting time. This sub-linear convergence rate is in fact too slow for any practical applications.

Let us introduce an artificial step-size adaptation rule that is used in the theoretical analysis of adaptive ES.

Definition 3.3 (Scale Invariance)**.** *Let x^* be the optimum of f, x_t and σ_t the parent and step-size of iteration t. If $\sigma_t = \sigma||x_t - x^*||$ for $\sigma > 0$ the step-size σ_t is called scale-invariant.*

Using scale invariant step size $(1+1)$ -ES reaches a convergence rate at most linear. This has been shown in different contexts by [Nekrutkin and Tikhomirov, 1993; Teytaud and Gelly, 2006; Jägersküpper, 2008]. This linear convergence rate is reached when the functions are spherical [Jebalia *et al.*, 2007].

One important argument is that proving linear convergence in convex quadratic functions is in fact a "weak" result, since there are other algorithms faster to solve convex quadratic functions. However, the results in the case of ES are not limited to convex quadratic functions. They include all strictly monotone transformation of the convex quadratic functions, which contains non-smooth, non-convex functions. Also, [Jebalia *et al.*, 2011] proves the robustness of the linear convergence in presence of noise when using scale invariant constant stepsize.

3.3.4 Evolution Strategies with reevaluation

The Evolutionary Algorithms are robust in rugged landscapes. And they are used effectively to optimize noisy functions [Beyer, 2001; Arnold and Beyer, 2006; Finck *et al.*, 2011]. Typically the algorithms are the same as the ones used on the noise-free setting, with some mechanism to cope with the noise. The reader can find in the literature examples of analysis of the population adaptation, the crossover or the selection process as mechanisms to deal with noise in an efficient way and the use of partial information approach [Friedrich *et al.*, 2015; Dang and Lehre, 2015; Prugel-Bennett *et al.*, 2015]. But the simplest method (and the one analysed in this thesis) is to reevaluate the points multiple times in order to reduce the effect of the noise. The number of reevaluations r_t can depend on parameters, the iteration (as we will explore in Chapter 5) or on the noise level [Hansen *et al.*, 2009].

Algorithm 3.8 (μ, λ)-ES algorithm with reevaluation r_t

Initialize Parent x_1, $t = 1$

1: **while** termination criterion not reach **do**
2: **for** $j = 1, \dots, \lambda$ **do**
3: Sample λ offsprings $P_\lambda^t = \{x_{t,1}, \dots, x_{t,\lambda}\}$ by $x_{t,j} = x_t + \sigma_t \mathcal{N}$ ▷ Mutation
4: Evaluate fitness r_t times and average $y_{t,j} = \frac{1}{r_t} \sum_{i=1}^{r_t} f(x_{t,j}, \omega_i)$ ▷ Evaluation
5: **end for**
6: Select from P_λ^t the μ best search points with largest fitness values ▷ Selection
7: Update x_t and σ_t using μ best search points and its fitness evaluations
8: $t = t + 1$
9: **end while**

Part II

Contributions

Chapter 4

General Iterative Noisy Optimization Algorithm: Regret Analysis

---

We study the performance of general iterative noisy optimization algorithms by proposing a general framework and a fundamental property for the algorithms. We prove that two algorithms using noisy function evaluations to approximate the gradient and the Hessian fit in the proposed framework and satisfy the fundamental property. These two algorithms can be viewed as gradient descent and Newton algorithm, except that they use noisy approximations instead of the real values of the gradient and Hessian.
We also prove convergences rates for the noisy Hessian based algorithm in terms of Simple and Cumulative Regret. We obtain that the modification of the parameters of the algorithm leads to different convergences rates. These converges rates coincide with the literature (on [Fabian, 1967; Shamir, 2013], [Dupac, 1957] and [Rolet and Teytaud, 2010a]) and we are also able to prove a conjecture presented in [Jebalia and Auger, 2008].
The results presented are based on the journal publication [Astete-Morales *et al.*, 2016a]. This is a joint work with Marie-Liesse Cauwet (TAO, INRIA Saclay), Jialin Liu (TAO, INRIA Saclay, now University of Essex) and Olivier Teytaud (TAO, INRIA Saclay, now Google).

4.1 Framework

We defined a black-box optimization algorithm in Section 2.2. In the case of noisy optimization algorithms, we need to add the fact that the function evaluations are perturbed by noise. In summary, at each iteration the algorithm chooses one or more search points and computes the noisy objective function value of each search point. Then the algorithm generates a recommendation based only on the noisy function values, without the use of gradient or higher order information.
The objective function f is corrupted by noise. The function f is such that $\mathbb{E}[f(x)]$ has a unique optimum x^*. We will consider three different *types of noise*. They will all be unbiased but their variance will vary depending on a parameter z and the Simple Regret. We consider then the variance of the noisy objective value to be:

$$\mathrm{Var}[f(x)] = O\left(\mathbb{E}[f(x) - f(x^*)]^z\right) \ , \qquad z \in \{0, 1, 2\} \ .$$

When $z = 0$ then the variance is *constant* all along the search space. When $z = 1$ and $z = 2$ the noise decreases in a *linear* and *quadratic* fashion when the Simple Regret decreases. We present in Algorithm 4.1 the Iterative Noisy Optimization Algorithm (INOA) that represents a general

framework for noisy optimization algorithms. It is provided with two methods: `SEARCH` and `OPT`. The method `SEARCH` outputs *search points* according to some rules. The method `OPT` outputs recommendations of the *optimum*. Both methods are only described in terms of their inputs and output. Therefore there are no limits for their internal behaviour as long as they use the input and output presented.

Algorithm 4.1 Iterative Noisy Optimization Algorithm (INOA)

Require: Step-size parameters $\alpha, A > 0$, Revaluation parameters $\beta, B > 0$, f, `SEARCH`, `OPT`

Initialize $\hat{x}_1$ randomly

for $t = 1, \dots, T$ **do**
  $\sigma_t = \frac{A}{t^\alpha}$
  $r_t = B\lceil t^\beta \rceil$
  for $n = 1, \dots, r_t$ **do**
    $x_{t,n} = \texttt{SEARCH}(\hat{x}_t, \sigma_t, n)$
    $y_{t,n} = f(x_{t,n}, \omega_t)$
  end for
  $\hat{x}_{t+1} = \texttt{OPT}(\hat{x}_t, (x_{t,n}, y_{t,n})_{n \in [1,\dots,r_t]})$
end for
Return $(\hat{x}_t)$

We will study the convergence of INOA in terms of Simple Regret (using the image of the recommendations $\hat{x}_t$) and Cumulative Regret (using the image of the search points $x_{t,n}$)[1]. In order to prove the convergence rates, we will prove before that INOA satisfies a fundamental property. This property is called Low Square Error (LSE) and defined in Definition 4.1. Any algorithm that is included in the framework and satisfies LSE is then affected by the consequences detailed on Section 4.4.

Definition 4.1 (Low Square Error (LSE))**.** *Let f be an objective function with noise variance defined by $z = 0, 1, 2$. Let r, σ be the parameters of an specific iteration of INOA. And let $\hat{x}$ be the output of* `OPT` *and x the point that serves as input for* `OPT`*. We say INOA satisfy LSE if the following condition is satisfied:*

$$||x - x^*|| = O(\sigma) \implies \mathbb{E}[||\hat{x} - x^*||^2] = O\left(\frac{\sigma^{2z-2}}{r}\right).$$

In sections 4.2 and 4.3 we will prove that the noisy gradient based algorithm and the noisy Hessian based algorithm are both included in the framework presented in Algorithm 4.1. Also, we prove that both algorithms satisfy the LSE.

As a side note, in this chapter we will use indistinctly the notation $O(\cdot)$ and the inequality that it represents, depending on the convenience of the notation.

[1] The algorithm computes sequences of search points, recommendation points and function evaluations, all indexed by the iteration t. Whenever we need to have a sequence indexed by the number of evaluations n we will abuse the notation and also use a sub-index identified by n instead of t. We will have a sequence of points $x_n = x_{t'}$ where $t' = \max\{t | \sum_{j=1}^{t-1} r_j \leq n\}$.

4.2 Noisy Gradient based Algorithm

In this section we establish `SEARCH` and `OPT` so that by plugging them into Algorithm 4.1 we obtain a noisy gradient based algorithm. The algorithm samples search points around the current recommendation of the optimum and uses these points to estimate the gradient and later obtain the next recommendation.

`SEARCH` will output search points from the set $S = S_+ \cup S_-$, where $S_+ = (x + \sigma e_j)_{j=1}^d$ and $S_- = (x - \sigma e_j)_{j=1}^d$. x is the former recommendation and $(e_j)_{j=1}^d$ represents the canonical orthonormal basis of $\mathbb{R}^d$. The set S has $2d$ elements and `SEARCH` outputs the n-th element of S when queried for the n-th time. As soon as $r > 2d$, `SEARCH` will repeat the elements belonging to S but INOA will evaluate these search points and give a different noisy function evaluation each time the search point is repeated. Note that S depends on σ, and that is the reason to declare σ as an argument in `SEARCH`.

`OPT` receives the former recommendation x and the r pairs of search point-noisy function evaluation that `SEARCH` outputs. It computes an estimate of the real function evaluation for each search point in S. The estimate will be simply the average among all the function evaluations on each search point. Afterwards, it will compute an estimate of the gradient by estimating each of its coordinates. And then using this estimated gradient to compute the next recommendation point $\hat{x}$. We define a set of function evaluations as follows:

$$Y_{j+} = \{\text{every evaluation of } x + \sigma e_j\}.$$
$$Y_{j-} = \{\text{every evaluation of } x - \sigma e_j\}.$$

For the notation we will use $|Y_{j+}|$ to denote the cardinality of Y_{j+} and $\sum Y_{j+}$ to denote the sum of all the elements of the set Y_{j+}.

In Algorithm 4.2 and 4.3 we can see in detail the `SEARCH` and `OPT` procedures for noisy gradient based Algorithm. We use a slightly different modulo function[2], only to maintain the correct sub-indexes.

Algorithm 4.2 `SEARCH` for noisy gradient based

procedure `SEARCH`(x, σ, n)
  $i = n \tilde{\bmod} 2d$
  $x_i =$ the i-th point of S
  Return (x_i)
end procedure

Algorithm 4.3 `OPT` for noisy gradient based

procedure `OPT`$(x, (x_i, y_i)_{i=1,\dots,r}, \sigma)$
  for $j = 1, \dots, d$ **do**
    $\hat{y}_{j+} = \frac{1}{|Y_{j+}|} \sum Y_{j+}$
    $\hat{y}_{j-} = \frac{1}{|Y_{j-}|} \sum Y_{j-}$
    $\hat{g}^{(j)} = \frac{\hat{y}_{j+} - \hat{y}_{j-}}{2\sigma}$
  end for
  $\hat{x} = x - \frac{1}{2}\hat{g}$
  Return $\hat{x}$
end procedure

[2]Definition: $i \tilde{\bmod} d = 1 + ((i-1) \bmod d)$

4.2.1 LSE property is satisfied for Noisy Gradient Based Algorithm

We consider a specific set of functions in order to prove that the noisy gradient based algorithm described by Algorithms 4.2 and 4.3 satisfies the LSE condition (Definition 4.1)
We will consider functions f such that their optimum is $x^* = 0$, and their expectation and variance satisfy Equations 4.1 and 4.2 respectively.

$$\mathbb{E}[f(x)] = ||x||^2, \tag{4.1}$$

$$\text{Var}[f(x)] = O\left(||x||^{2z}\right), \qquad z \in \{0, 1, 2\}. \tag{4.2}$$

We know that (see property in Chapter 2, Lemma 2.3):

$$\mathbb{E}[||\hat{x}||^2] = \sum_{j=1}^{d} \left(\text{Var}[\hat{x}^{(j)}] + (\mathbb{E}[\hat{x}^{(j)})]^2\right).$$

Then, by the definition of $\hat{g}$ in Algorithm 4.3 and using 4.1 we obtain[3]:

$$\mathbb{E}[\hat{g}^{(j)}] = 2x^{(j)}. \tag{4.3}$$

Then, $\mathbb{E}[\hat{x}^{(j)}] = 0$ using $\hat{x}$ in Algorithm 4.3.
Let x such that $||x|| \leq O(\sigma)$. Note that either $|Y_{j+}| = |Y_{j-}| = r/2$ or $|Y_{j+}| = \lceil r/2 \rceil$ and $|Y_{j-}| = \lfloor r/2 \rfloor$. The occurance of the former two cases depends on the relationship between j and r. Either way, in this case the only concerned is about the order of $|Y_{j+}|$ and $|Y_{j-}|$. Using the former argument and Equation 4.2 we obtain[4]:

$$\mathbb{E}[||\hat{x}||^2] = O\left(\frac{\sigma^{2z-2}}{r}\right). \tag{4.4}$$

Therefore we have proved that the noisy gradient based Algorithm satisfies the LSE condition for f that satisfies 4.1 and 4.2.

4.3 Noisy Hessian based Algorithm

In this section we establish `SEARCH` and `OPT` so that by plugging them into Algorithm 4.1 we obtain a noisy Hessian based algorithm. The principle is the same as the one explained for the noisy gradient based algorithm, in Section 4.2. `SEARCH` will output search points from an extended set, in comparison to the set S used for noisy gradient based algorithm. Let $\mathfrak{S} = S \cup \{x \pm \sigma e_i \pm \sigma e_j; 1 \leq i < j \leq d\}$ with cardinality $|\mathfrak{S}| = 2d^2$. We extend naturally the definition for the sets of function evaluations Y_{j+} and Y_{j-} as follows:

$$Y_{j+,k+} = \{\text{every evaluation of } x + \sigma e_j + \sigma e_k\}.$$
$$Y_{j-,k+} = \{\text{every evaluation of } x - \sigma e_j - \sigma e_k\}.$$
$$Y_{j+,k-} = \{\text{every evaluation of } x + \sigma e_j - \sigma e_k\}.$$
$$Y_{j-,k-} = \{\text{every evaluation of } x - \sigma e_j - \sigma e_k\}.$$

[3] See details in Appendix, Section 10.1.
[4] See details in Appendix, Section 10.1.

Algorithm 4.4 `SEARCH` for noisy Hessian based

1: **procedure** `SEARCH`(x, σ, n)
2: $i = n \tilde{\mod} 2d^2$
3: $x_i =$ the i-th point of $\mathfrak{S}$
4: **Return** (x_i)
5: **end procedure**

Algorithm 4.5 `OPT` for noisy Hessian based

1: **procedure** `OPT`$(x, (x_i, y_i)_{i=1,\dots,r}, \sigma, c_0)$
2: **for** $j = 1, \dots, d$ **do**
3: $\hat{y}_{j+} = \frac{1}{|Y_{j+}|} \sum Y_{j+}$
4: $\hat{y}_{j-} = \frac{1}{|Y_{j-}|} \sum Y_{j-}$
5: $\hat{g}^{(j)} = \frac{\hat{y}_{j+} - \hat{y}_{j-}}{2\sigma}$
6: **end for**
7: **for** $1 \leq j, k \leq d$ **do**
8: $\hat{y}_{j+,k+} = \frac{1}{|Y_{j+,k+}|} \sum Y_{j+,k+}$
9: $\hat{y}_{j-,k+} = \frac{1}{|Y_{j-,k+}|} \sum Y_{j-,k+}$
10: $\hat{y}_{j+,k-} = \frac{1}{|Y_{j+,k-}|} \sum Y_{j+,k-}$
11: $\hat{y}_{j-,k-} = \frac{1}{|Y_{j-,k-}|} \sum Y_{j-,k-}$
12: $\hat{h}^{(j,k)} = \frac{(\hat{y}_{j+,k+} - \hat{y}_{j-,k+}) - (\hat{y}_{j+,k-} - \hat{y}_{j-,k-})}{4\sigma^2}$
13: **end for**
14: $\hat{h} = \frac{\hat{h} + t(\hat{h})}{2}$
15: **if** $\hat{h}$ positive definite with smallest eigenvalue greater than c_0 **then**
16: $\hat{x} = x - \hat{h}^{-1}\hat{g}$
17: **else**
18: $\hat{x} = x$
19: **end if**
20: **Return** $\hat{x}$
21: **end procedure**

4.3.1 LSE property is satisfied for noisy Hessian based Algorithm

We consider a specific set of functions in order to prove that the Hessian based algorithm described by Algorithms 4.4 and 4.5 satisfies the LSE condition (Definition 4.1).

We will consider functions such that the expectation and the variance satisfy the Equations 4.5 and 4.6 respectively.

$$\mathbb{E}_\omega[f(x,\omega)] = \sum_{1 \leq j,k \leq d} a_{j,k} x^{(j)} x^{(k)} + \sum_{1 \leq j,k,l \leq d} b_{j,k,l} x^{(j)} x^{(k)} x^{(l)} + O(||x||^3), \quad \text{with } a_{j,k} = a_{k,j}. \tag{4.5}$$

$$\mathrm{Var}[f(x,\omega)] = O\left(||x||^{2z}\right), \qquad z \in \{0,1,2\}. \tag{4.6}$$

In the Appendix, Section 10.2, the reader can find the proof of the following property: Let h be the Hessian of $\mathbb{E}[f]$ at 0. Denote $h = (2c_{j,k})_{1 \leq j,k \leq d}$. Assume $\exists c_0 > 0$ such that h is positive definite with least eigenvalue greater than $2c_0$. Then we know that there exists $\sigma_0 > 0, K > 0, C > 0$ such

that for all σ that satisfies $\sigma < \sigma_0$ and $\sigma^{6-2z} < \frac{K}{r}$. Then for all x such that $||x|| \leq C\sigma$ we have:

$$\mathbb{E}[||\hat{x}||^2] = O\left(\frac{\sigma^{2z-2}}{r}\right),$$

where $\hat{x}$ is the output of `OPT`, x the input of `OPT`.

Remark 4.1. *In order to get the result above we need to assume* $\alpha(6-2z) - \beta > 0$.

4.4 Convergence Rates

In this section we study the convergence rate of INOA with the (`SEARCH`, `OPT`) process as in Algorithms 4.4 and 4.5

We start by proving two propositions (Propositions 4.1 and 4.2). Then we prove the convergence in Theorem 4.1 using the fact that the hypothesis from the latter propositions are met in our case. We use the consequences of Propositions 4.1 and 4.2 to prove Theorem 4.2: the exact form of the convergence rate for SR and CR, that depends on the parametrization of the algorithm.

Proposition 4.1. *If* (`SEARCH`, `OPT`) *satisfy LSE and* $\alpha(2z-4) + \beta > 1$ *and* $||\hat{x}_n|| = O(\sigma_n)$ *with probability* $1-\delta$. *Then* $\exists M > 0$ *such that:*

$$\mathbb{E}||\hat{x}_{n+1}||^2 \leq M(n+1)^{-\alpha(2z-2)-\beta}.$$

Proof. Since $||\hat{x}_n|| = O(\sigma_n)$ with probability $1-\delta$, using LSE we obtain:

$$\begin{aligned}
\mathbb{E}[||\hat{x}_{n+1}||^2] &\leq O\left(\frac{\sigma_n^{2z-2}}{r_n}\right), \\
&= O\left(\left(\frac{A}{n^\alpha}\right)^{2z-2} \frac{1}{B\lceil n^\beta \rceil}\right) && \text{by Def. of } r_n, \sigma_n \text{ in Alg 4.1}, \\
&\leq O\left(\frac{A^{2z-2}}{B} n^{-\alpha(2z-2)-\beta} \frac{(n+1)^{-\alpha(2z-2)-\beta}}{(n+1)^{-\alpha(2z-2)-\beta}}\right), \\
&= O\left(\frac{A^{2z-2}}{B}\left(\frac{n}{n+1}\right)^{-\alpha(2z-2)-\beta} (n+1)^{-\alpha(2z-2)-\beta}\right), \\
&\leq \underbrace{L\frac{A^{2z-2}}{B} \sup_{n\geq 1}\left(\frac{n}{n+1}\right)^{-\alpha(2z-2)-\beta}}_{:=M} (n+1)^{-\alpha(2z-2)-\beta} && \text{some } L > 0\,. \qquad (4.7)
\end{aligned}$$

Note that $\sup_{n\geq 1}\left(\frac{n}{n+1}\right)^{-\alpha(2z-2)-\beta} < \infty$ so the have the result. □

Proposition 4.2. *If* `SEARCH` *is a procedure as in Algorithm 4.4 and* $||\hat{x}_n|| \leq K\sigma_n$, *then:*

$$||x_{n,i}|| \leq (K+2)\sigma_n.$$

Proof. `SEARCH` outputs each new search point at a distance at most $2\sigma_n$ from the provided point $\hat{x}_n$. Using $||\hat{x}_n|| \leq K\sigma_n$ we obtain:

$$||x_{n,i}|| \leq ||x_{n,i} - \hat{x}_n|| + ||\hat{x}_n|| \leq (K+2)\sigma_n.$$

□

Theorem 4.1. *Let f with noise $z \in \{0,1,2\}$ be the objective function. Consider the algorithm 4.1 with parameters A, B, α, β with B large enough and*

$$\alpha(2z-4)+\beta > 1. \tag{4.8}$$

Assume that INOA with `SEARCH` *and* `OPT` *defined in Algorithms 4.4 and 4.5 satisfy LSE (definition 4.1) for f. Let $\hat{x}_1$ such that $||\hat{x}_1|| \leq KA$, for some $K > 0$. Then for all n, $||\hat{x}_n|| \leq K\sigma_n$ with probability $1-\delta$, δ small enough.*

Proof. We will use induction. Let H_n be the inductive hypothesis: For any $1 \leq i \leq n$, $||\hat{x}_n|| \leq K\sigma_n$ with probability $1-\delta_n$, with δ_n defined as follows:

$$\delta_n = \underbrace{\frac{\delta}{\sum_{i=1}^{\infty} i^{-\alpha(2z-4)-\beta}}}_{:=c(\delta)} \sum_{i=1}^{n} i^{-\alpha(2z-4)-\beta}.$$

Note that δ_n is well defined and $\delta_n \leq \delta$ .We have used the fact that $\sum_{i=1}^{\infty} i^{-\alpha(2z-4)-\beta} < \infty$ by hypothesis 4.8. We prove now the inductive steps:

- H_1: By hypothesis $\hat{x}_1 \leq KA = K\sigma_1$ using the definition of σ_t in Algorithm 4.1. Therefore H_1 holds.
- $H_n \implies H_{n+1}$: Using Markov's inequality (Lemma 2.5) we obtain:

$$\begin{aligned}\mathbb{P}[||\hat{x}_{n+1}|| > K\sigma_{n+1}] &= \mathbb{P}[||\hat{x}_{n+1}||^2 > K^2\sigma_{n+1}^2] \ , \\ &\leq \frac{\mathbb{E}[||\hat{x}_{n+1}||^2]}{K^2\sigma_{n+1}^2}. \end{aligned} \tag{4.9}$$

  By inductive hypothesis, H_n is true. That is to say $||\hat{x}_n|| \leq K\sigma_n$ with probability $1-\delta_n$. Then we can use Proposition 4.1 to bound $E[||\hat{x}_{n+1}||^2]$ using the relationship in 4.7 and we obtain:

$$\begin{aligned}\mathbb{P}[||\hat{x}_{n+1}|| > K\sigma_{n+1}] &\leq \frac{M}{K^2\sigma_{n+1}^2}(n+1)^{-\alpha(2z-2)-\beta} && M \text{ defined in eq. 4.7 ,} \\ &\leq \frac{M}{K^2A^2c(\delta)}c(\delta)(n+1)^{-\alpha(2z-4)-\beta} \ , \\ &=: \varepsilon_n.\end{aligned}$$

  For all $B > B_0 = \frac{KA^{2z-4}}{C^2c(\delta)} \sup_{n\geq 1}\left(\frac{n}{n+1}\right)^{-\alpha(2z-2)-\beta}$ we can ensure that $\frac{M}{K^2A^2c(\delta)} \leq 1$. By hyphotesis $-\alpha(2z-4)-\beta < -1$, therefore $\varepsilon_n \leq 1$. Then $||\hat{x}_{n+1}|| \leq K\sigma_{n+1}$ with probability $(1-\delta_n)(1-\varepsilon_n)$, which implies that:

$$||\hat{x}_{n+1}|| \leq K\sigma_{n+1} \qquad \text{with probability at least } 1-\delta_n-\varepsilon_n = 1-\delta_{n+1}. \tag{4.10}$$

  With 4.10 we have proved H_{n+1} and the proof of the theorem is complete.

□

Theorem 4.2. *Assume Algorithm 4.1 satisfies LSE for some f with noise $z \in \{0, 1, 2\}$. Assume* (SEARCH, OPT) *included in 4.1 as in Algorithms 4.4 and 4.5 and that f is such that:*

$$\mathbb{E}_\omega[f(x, \omega) - f(x^*, \omega)] = ||x - x^*||^2. \tag{4.11}$$

Then the convergence rates of SR and CR are as follows:

$$s(SR) = \frac{-\alpha(2z - 2) - \beta}{\beta + 1}, \tag{4.12}$$

$$s(CR) = \frac{\max(0, 1 + \beta - 2\alpha)}{\beta + 1}. \tag{4.13}$$

Proof. First note that the number of evaluations until iteration n is $\sum_{i=1}^{n} r_n = O(n^{\beta+1})$. Remember that SR_n corresponds to the $1 - \delta$ quantile of $||\hat{x}_n - x^*||^2$ by using the hypothesis of Equation 4.11. To prove 4.12 we use first Markov's inequality:

$$\mathbb{P}\left[||x - x^*|| > \frac{\mathbb{E}||x - x^*||^2}{\delta}\right] < \delta.$$

Using Proposition 4.1 we get that:

$$\mathbb{E}[||x - x^*||^2] = O(n^{-\alpha(2z-2)-\beta}).$$

Therefore the $1 - \delta$ quantile of $||\hat{x}_n - x^*||^2 = O(n^{-\alpha(2z-2)-\beta})$. Using the number of evaluations until iteration n we obtain the result in Equation 4.12.
To prove Equation 4.13 remember that CR_n is the $1-\delta$ quantile of $\sum_{1 \le i,m \le n} \mathbb{E}f(x_{i,m}, \omega) - f(x^*, \omega)$ and given the way search points are chosen at each iteration.

$$\begin{aligned}
\sum_{1 \le i,m \le n}^{n} \mathbb{E}[f(x_{i,m}, \omega) - f(x^*, \omega)] &= \sum_{i}^{n} r_i O(||x_{i,1} - x^*||^2) \ , \\
&= \sum_{i}^{n} (C + 2) i^{\beta - 2\alpha} && \text{by Prop. 4.2, Def. of } \sigma_n \ , \\
&= \begin{cases} O(n^{-2\alpha+\beta+1}) & \text{if } -2\alpha + \beta > -1 \ , \\ O(\log(n)) & \text{if } -2\alpha + \beta = -1 \ , \\ O(1) & \text{otherwise.} \end{cases}
\end{aligned}$$

Using the number of evaluations until iteration n we obtain the result in Equation 4.13.

□

4.5 Convergence rates for Noisy Hessian based Algorithm: comparison with the State of the Art

We will consider the noisy Hessian based Algorithm and obtain its convergence rates for SR and CR. We will consider smooth functions with at least two derivatives. We assume two conditions on the parameters:

- $\alpha(6-2z)-\beta>0$ so that the noisy Hessian based Algorithm satisfies LSE. Except in the case of $z=0$ and quadratic functions, since it is not necessary.

- $\alpha(2z-4)+\beta>1$ so that we can verify Theorem 4.1. Except in the case of $z=0$ and quadratic functions, it is sufficient to have: $-4\alpha+\beta>1$.

In the case of $z=0,1,2$ we obtain the following restrictions, considering $\alpha,\beta>0$

$$\alpha(6-2z)-\beta>0=\begin{cases}6\alpha-\beta>0 & z=0\ ,\\ 4\alpha-\beta>0 & z=1\ ,\\ 2\alpha-\beta>0 & z=2\ .\end{cases} \tag{4.14}$$

$$\alpha(2z-4)+\beta>1=\begin{cases}-4\alpha+\beta>1 & z=0\ ,\\ -2\alpha+\beta>1 & z=1\ ,\\ \beta>1 & z=2\ .\end{cases} \tag{4.15}$$

Now we use the Theorem 4.2 to compute the convergence rate for SR and CR, through Equations 4.12 and 4.13. Note that only the convergence rate of SR (Equation 4.12) depends on z. The difference between the cases will become from the restrictions described in 4.14 and 4.15. Tables 4.1, 4.2 and 4.3 show the minimization problems to optimize SR (left column) and CR (right column). The first row presents the minimization problem deduced from Proposition 4.2. The second row presents the solution for SR and CR. We present the value of the minimized regret on the respective case and also the value of the regret that is not being minimized.

Minimize SR	Minimize CR
$\min_{\alpha,\beta}\frac{2\alpha-\beta}{\beta+1}$	$\min_{\alpha,\beta}\frac{\max(0,-2\alpha+\beta+1)}{\beta+1}$
if α fix and $\beta\to\infty \implies \begin{cases}s(SR)\to-1\\ s(CR)\to1\end{cases}$	if $\alpha\to\infty$ and $\beta=4\alpha+1^+ \implies \begin{cases}s(CR)\to1/2\\ s(SR)\to-1/2\end{cases}$

Table 4.1: Noise type $z=0$

Minimize SR	Minimize CR
$\min_{\alpha,\beta}\frac{-\beta}{\beta+1}$	$\min_{\alpha,\beta}\frac{\max(0,-2\alpha+\beta+1)}{\beta+1}$
if α fix and $\beta\to\infty \implies \begin{cases}s(SR)\to-1\\ s(CR)\to1\end{cases}$	if $\alpha\to\infty$ and $\beta=4\alpha+1^+ \implies \begin{cases}s(CR)\to0\\ s(SR)\to-1\end{cases}$

Table 4.2: Noise type $z=1$

Minimize SR	Minimize CR
$\min_{\alpha,\beta} \frac{-2\alpha-\beta}{\beta+1}$	$\min_{\alpha,\beta} \frac{\max(0,-2\alpha+\beta+1)}{\beta+1}$
if α fix and $\beta \to \infty \implies \begin{cases} s(SR) \to -\infty \\ s(CR) \to 0 \end{cases}$	if $\alpha \to \infty$ and $\beta = 4\alpha + 1^+ \implies \begin{cases} s(CR) \to 0 \\ s(SR) \to -\infty \end{cases}$

Table 4.3: Noise type $z = 2$

Table 4.4 presents a summary of the results from tables 4.1, 4.2 and 4.3. Note that we do not obtain necessarily the best SR and CR simultaneously. Except in the case $z = 2$ where we obtain optimal results for both SR and CR. This can be justified due to the "low" influence of the noise with $z = 2$.

z	Parametrization	$s(SR)$	$s(CR)$	Optimality	Reference Literature
0	$\alpha > 0$, $\beta \to \infty$	-1	1	Optimal SR	[Fabian, 1967; Shamir, 2013]
0	$\alpha \to \infty$, $\beta = 4\alpha + 1^+$	$-1/2$	$1/2$	Not optimal	
0	$\alpha = \beta/6$, $\beta \to \infty$	$-2/3$	$2/3$	Not optimal	[Dupac, 1957]
1	$\alpha = \beta^+/4$, $\beta \to \infty$	-1	1	Not known	[Rolet and Teytaud, 2010a]
1	$\alpha \to \infty$, $\beta = 2\alpha + 1^+$	-1	0	Not known	
2	$\alpha \to \infty$, $\beta > 1$	$-\infty$	0	Optimal SR and CR	[Jebalia and Auger, 2008] [Conjecture]

Table 4.4: Results different parametrizations optimizing SR and CR

4.6 Conclusion

In this chapter we present a general framework for noisy optimization algorithms called *Iterative Noisy Optimization Algorithm* (INOA) in Algorithm 4.1. This framework represents a big spectrum of algorithms used in noisy optimization and it is basically divided in two big procedures: `SEARCH` and `OPT`. The procedure `SEARCH` takes care of the generation of *search points* while `OPT` outputs in each iteration a *recommendation to the optimum*.

We also propose a general property for algorithms: the *Low Square Error* (LSE). The name comes from the fact that we limit the distance between the recommendations and the optimum if the search points are sufficiently close.

We prove in particular that we include the noisy gradient based (in Section 4.2) and the noisy Hessian based algorithm (in Section 4.3) as part of the algorithms described by the framework on Algorithm 4.1 by writing in detail the procedures `SEARCH` and `OPT`. We prove also that both algorithms satisfy the LSE.

Finally, in Section 4.4 we prove the convergences rate of INOA, with `SEARCH` procedure as in the noisy Hessian based algorithm. We observe the Simple and Cumulative Regret. In Section 4.5 we summarize the results and compare them with the literature. We observe that, depending on the parametrization of the algorithm, we can reproduce several results obtain in the literature.

Even more, we obtain a new result, conjectured previously in the literature. The reader can see Table 4.4.

The fact that the results depends on the parametrisation of the algorithm is a disadvantage. When solving problems in the real world, we rarely have the opportunity to have information over the objective function measurements such as the "type" of noise we could be dealing with. Nonetheless, the remarkable result is that we can obtain a wide range of convergence rates using only one algorithm, and analysing its parameters.

Chapter 5

Log-log convergence of Evolution Strategies

In Chapter 4 we studied algorithms of the Linesearch family. In this chapter we step aside from those type of algorithms and study Evolutionary Algorithms. We prove that a particular kind of Evolutionary Algorithms, the Evolution Strategies converge for noisy optimization problems. We use reevaluation to mitigate the noise and a convergence result for Evolution Strategies on noise-free optimization [Auger, 2005].
We present theoretical analyses for two types of reevaluation schemes: exponential and adaptive. We also consider scale invariance for the analysis of the exponential scheme. We ignore the scale variance and replace it by another property in the analysis of the adaptive scheme. We obtain that the order of convergence for Simple Regret is log-log in both cases. We also present experiments for a polynomial scheme of reevaluation, confirming the results obtained theoretically.
The results presented are based on the conference publication [Astete-Morales *et al.*, 2014]. This is a joint work with Jialin Liu (TAO, INRIA Saclay, now University of Essex) and Olivier Teytaud (TAO, INRIA Saclay, now Google).

5.1 Preliminaries

In this section we present the three basic elements that constitute the results of this work. First, we start by analysing the convergence result from [Auger, 2005]. We develop the results into the precise form we will use to prove the convergence of the Evolution Strategy for noisy problems. Second, we present the three different reevaluation schemes we will use in the analysis: exponential, adaptive and polynomial. And third, we present the objective (or fitness) function we will use for the convergence analysis in this chapter.

5.1.1 Noise Free Case

From the analysis in Theorem 4, [Auger, 2005] over the $(1, \lambda)$-SA-ES we know that considering $F(x) = ||x||^2$ the following results are satisfied almost surely:

$$\frac{1}{n} \log(||x_n||) \xrightarrow{n\to\infty} R, \tag{5.1}$$

$$\frac{1}{n} \log(\sigma_n) \xrightarrow{n\to\infty} R. \tag{5.2}$$

If we assume R is negative, then the Equation 5.1 means that the sequence of recommendations of $(1, \lambda)$-SA-ES converges to the optimum in a log-linear scale.
We can write the Equation 5.1 equivalently, for all $\varepsilon > 0$, and n sufficiently large

$$-\varepsilon < \underbrace{\frac{1}{n}\log(||x_n||) - R < \varepsilon}_{(\star)}\,. \tag{5.3}$$

Then,

$$\begin{aligned}
(\star) &\implies \frac{1}{n}\log(||x_n||) \leq U && \forall U \text{ such that } R < U\ , \\
&\implies \underbrace{\log(||x_n||) - Un}_{:=A_n} \leq 0\ , \\
&\therefore \sup_{n\geq 1} A_n \leq 0.
\end{aligned}$$

Let $Y = \exp(\sup_{n\geq 1} A_n)$ and Q the $1 - \delta/4$ quantile of Y. Therefore, with probability $1 - \delta/4$:

$$\begin{aligned}
&\exp(\sup_{n\geq 1} A_n) \leq Q \\
\implies &\exp(A_n) \leq Q \quad \forall n && \exp(\cdot) \text{ is an increasing function} \\
\implies &||x_n|| \leq Q\exp(Un) && \text{using the definition of } A_n.
\end{aligned}$$

We obtain an analogous lower bound for $||x_n||$ using the left hand inequality in 5.3 as follows. We obtain that $B_n := \log(||x_n||) - Ln \leq 0$, $\forall L$ such that $R > L$. Then we define $Z = \exp(\inf_{n\geq 1} B_n)$ and q the $\delta/4$ quantile of Z. Therefore, with probability $1 - \delta/4$:

$$\begin{aligned}
&\exp(\inf_{n\geq 1} B_n) \geq q \\
\implies &\exp(B_n) \geq q \quad \forall n \\
\implies &||x_n|| \geq q\exp(Ln).
\end{aligned}$$

We can use the same arguments to obtain upper and lower bounds for σ_n, starting from inequalities on (5.2). In summary we obtain that for any U, L such that $L < R < U$, there exists Q, q, Q_σ, $q_\sigma > 0$ such that with probability at least $1 - \delta$, for all n sufficiently large:

$$q\exp(Ln) \leq ||x_n|| \leq Q\exp(Un), \tag{5.4}$$

$$q_\sigma\exp(Ln) \leq \sigma_n \leq Q_\sigma\exp(Un). \tag{5.5}$$

Note that the bounds on 5.4 and 5.5 are interesting only when R is strictly negative, so that we can choose L and U strictly negative as well.

5.1.2 Algorithm (μ, λ)-ES with reevaluations

We analyze Algorithm 3.8 with different types of reevaluation schemes r_t: exponential, adaptive and polynomial. We abuse the notation and use always K and η for every type of reevaluation

scheme, but they do not have to be the same for all three. We also consider the ceiling function of r_t so we can ensure $r_t \in \mathbb{N}$ for all t.

Type	Notation	r_t
exponential	$r^{\exp}_{K\eta}$	$K\eta^t$
adaptive	$r^{\text{adap}}_{K\eta}$	$K\sigma_t^{-\eta}$
polynomial	$r^{\text{poly}}_{K\eta}$	Kt^{η}

Table 5.1: Reevaluation schemes r_t

The (μ, λ)-ES with reevaluation scheme $r^{\exp}_{K\eta}$ will be called (μ, λ)-ES-$r^{\exp}_{K\eta}$. Idem definition for adaptive and polynomial reevaluation schemes. Note that (μ, λ)-ES-$r^{\exp}_{K\eta}$ preserves the same properties as (μ, λ)-ES with regards to the x_t and σ_t. The only difference is the amount of reevaluations made at each iteration.

5.1.3 Fitness function

We will consider the p *noisy function* defined for some $p > 0$.

$$\begin{aligned} f : \mathbb{R}^d \times \Omega &\to \mathbb{R} \\ (x, \omega) &\mapsto ||x||^p + \omega, \quad \text{with } \omega \sim \mathcal{N}, \end{aligned} \tag{5.6}$$

where $\mathcal{N}$ is either a Gaussian or a bounded variance distribution. Note that for $p = 2$ we obtain the noisy sphere function, for which we have the results on the noise-free case detailed in Section 5.1.1.

5.2 Non adaptive exponential reevaluation and scale invariance

We will prove that if (μ, λ)-ES with scale invariance converges in the noise-free case, then it also converges in the noisy case, considering (μ, λ)-ES with exponential reevaluation when the parameters are large enough. We obtain the same log-linear convergence in terms of number of iterations, but once we take into account the extra number of evaluations done by (μ, λ)-ES-$r^{\exp}_{K\eta}$, we observe log-log convergence.

Theorem 5.1. *Assume (μ, λ)-ES solving the sphere function satisfies:*

1. *For some $L, U < 0$, for any $\delta > 0$ $\exists q, Q$ such that with probability $1 - \delta/2$ for all t sufficiently large,*
$$q \exp(Lt) \leq ||x_t|| \leq Q \exp(Ut). \tag{5.7}$$
2. *Scale invariance: For all t,*
$$\sigma_t = C||x_t||. \tag{5.8}$$

Then for any $\delta > 0$, there is $K_0, \eta_0 > 0$ such that for all $K \geq K_0$, $\eta \geq \eta_0$, the points x_t output by (μ, λ)-ES-$r^{exp}_{K\eta}$ solving the p noisy function satisfies 5.7 with probability $1 - \delta$.

Proof. We will prove that the probability of a misranking is small, provided that there are enough evaluations per search point. Let x, $y \in \mathcal{D}$ be two search points output by (μ, λ)-ES-$r^{\exp}_{K\eta}$. A misranking occurs when the real fitness values and the noisy fitness values are ordered in "contradictory" way on the fitness space. That is to say, the expressions in 5.9 and 5.10 occur simultaneously.

$$F(x) \leq F(y) \iff ||x||^p \leq ||y||^p, \tag{5.9}$$

$$f(x) \geq f(y) \iff ||x||^p + z_x \geq ||y||^p + z_y \ . \tag{5.10}$$

The misranking happens in two situations. One, when x and y have very similar fitness values, so even a small amount of noise can invert their real order on the fitness space. Two, when the noise is so large that it also alters the order.

We start by bounding the probability of the fitness to be too similar. Consider $\delta_t = \exp(-\gamma t)$ for some fixed $\gamma > 0$ and x,y be two search points at iteration $t+1$. Define $p^f_t = \mathbb{P}(|||x||^p - ||y||^p| \leq \delta_t)$. Since x and y are two search points at iteration $t+1$, they are mutated offsprings of parent x_t. Therefore $x = x_t + \sigma z_x$ and $y = x_t + \sigma z_y$ where z_x and z_y represent the realization of the random variables included in the mutation that generates x and y respectively. Then,

$$\begin{aligned}
p^f_t &= \mathbb{P}(|||x_t + \sigma_t z_x||^p - ||x_t + \sigma_t z_y||^p| \leq \delta_t) \ , \\
&= \mathbb{P}\left(|||1 + Cz_x||^p - ||1 + Cz_y||^p| \leq \frac{\delta_t}{||x_t||^p}\right) && \text{using hypothesis 5.8 ,} \\
&\leq \mathbb{P}\left(|||1 + Cz_x||^p - ||1 + Cz_y||^p| \leq \frac{\exp(-\gamma t)}{(q\exp(Lt))^p}\right) && \text{using hypothesis 5.7 ,} \\
&\leq Mq^{-p}\exp((-Lp - \gamma)t) && \text{using } z_x, z_y \sim \mathcal{N} \ .
\end{aligned} \tag{5.11}$$

In Inequality 5.11, M represents the maximum of the density of $|||1 + Cz_x||^p - ||1 + Cz_y||^p|$. Note that $M \in \mathbb{R}$ is a consequence of the fact that z_x and z_y are i.i.d. Gaussian random variables. We can bound the probability of any pair search points to have very similar fitness value in any iteration t denoted by P^f_t, obtaining $P^f_t \leq \lambda^2 p^f_t$.

Now we will bound the probability of the noise being too large. We will denote this probability by p^z_t. More specifically, the noise is large if the estimation of the real fitness value using the noisy fitness values is larger than $\delta_t/2$. Given that the estimation uses $K\eta^t$ reevaluations of each point to compute a mean of the noisy fitness values of that point, we obtain:

$$\begin{aligned}
p^z_t &= \mathbb{P}\left[\left|\frac{\mathcal{N}}{\sqrt{K\eta^t}}\right| \geq \frac{\delta_t}{2}\right] \ , \\
&= \mathbb{P}\left[\mathcal{N} \geq \frac{\delta_t}{2}\sqrt{K\eta^t}\right] \ , \\
&\leq \frac{4}{K}\frac{1}{\delta_t^2\eta^t} && \text{using Chebyshev, Lemma 2.6 ,} \\
&= \frac{4}{K}\exp((2\gamma - \log(\eta))t) \ .
\end{aligned} \tag{5.12}$$

Let P^z_t denote the probability of the noise being too large for at least one offspring at any iteration t. Then, $P^z_t \leq \lambda p^z_t$.

Using 5.11 and 5.12 we can bound the probability of misranking, denoted by P_t, as follows:

$$\begin{aligned} P_t &\leq P_t^f + P_t^z \ , \\ &\leq \lambda^2 p_t^f + \lambda p_t^z \ , \\ &\leq \lambda^2 M q^{-p} \exp((-Lp - \gamma)t) + \lambda \frac{4}{K} \exp((2\gamma - \log(\eta))t) \ . \end{aligned} \tag{5.13}$$

The latter is a bound for the probability of misranking at iteration t. If the parameters are adequate in 5.13 we can bound the probability of misranking in the whole process of the ES arbitrarily $\sum_{t \geq 1} P_t^z \leq \delta$, provided that γ, η and K are large enough.

In summary, we obtain that with probability at least $1 - \delta$, the algorithm (μ, λ)-ES-$r^{\exp}_{K\eta}$ obtains exactly the same rankings as (μ, λ)-ES, and therefore the sequence x_t output by (μ, λ)-ES-$r^{\exp}_{K\eta}$ solving the p noisy function maintains the property 5.7 assumed for the output of (μ, λ)-ES on the noise-free setting. □

Corollary 5.1. *Let $n(t)$ be the number of evaluations at the end of iteration t. Then, with probability at least $1 - \delta$*

$$\frac{\log(||x_n||)}{\log(n)} \to -\frac{R}{\log \eta} \ . \tag{5.14}$$

Proof. First we note that $n(t) = K\eta \frac{\eta^t - 1}{\eta - 1}$. Then we use 5.7 and the fact that we can choose $L = R - \varepsilon$ and $U = R + \varepsilon$ for all $\varepsilon > 0$. □

5.3 Adaptive scale dependent reevaluation

In Section 5.2 we have used a scale invariance assumption represented by Equation 5.8. This feature is not realistic, since it demands the knowledge in advance of the distance to the optimum. This section presents analogous results to the ones in Section 5.2 but only using the assumption of log-linear convergence on the noise-free case. We know that this is possible, thanks to the result in [Auger, 2005], and its consequences detailed in Section 5.1.1. We also change the exponential reevaluation and use an adaptive scheme, depending on the stepsize.

Theorem 5.2. *Assume (μ, σ)-ES solving the sphere function satisfies*

1. *For some $L, U > 0$, for any $\delta > 0$ $\exists q, Q$ such that with probability $1 - \delta/2$ for all t sufficiently large:*

$$q \exp(Lt) \leq ||x_t|| \leq Q \exp(Ut) \ , \tag{5.15}$$

$$q_\sigma \exp(Lt) \leq \sigma_t \leq Q_\sigma \exp(Ut) \ . \tag{5.16}$$

$$\tag{5.17}$$

2. *Assume that the number of reevaluations per iteration is*

$$K \left(\frac{1}{\sigma_t} \right)^\eta \ .$$

Then, for any $\delta > 0$, there is $K_0, \eta_0 > 0$ such that for all $K \geq K_0$, $\eta \geq \eta_0$, the points x_t output by $r^{adap}_{K\eta}$-(μ, λ)-ES with bounded density mutation solving the p noisy function satisfy 5.15 with probability $1 - \delta$

Proof. The proof is very similar to the one of Theorem 5.1. We only need to adapt some steps of the reasoning.
We do not have here scale invariance or the distribution of the random variable associated to the mutation. Therefore:

$$\begin{aligned} p^f_t &= \mathbb{P}(|||x_t + \sigma_t z_x||^p - ||x_t + \sigma_t z_y||^p| \leq \delta_t) \ , \\ &= \mathbb{P}\left(|||1 + C_t z_x||^p - ||1 + C_t z_y||^p| \leq \frac{\delta_t}{||x_t||^p}\right) \ , \\ &\leq \frac{1}{C^d_t} M q^{-p} \exp((-Lp - \gamma)t) \qquad \text{using Lemma 2.4,} \end{aligned}$$

where $C_t = \sigma_t / ||x_t|| > 0$ and M is the maximum of the density of $|||1 + C_t z_x||^p - ||1 + C_t z_y||^p|$. We know M is bounded because of the hypothesis on the density of the mutation random variable. We will prove now that the number of function evaluations for each iteration is larger than in Theorem 5.1. We have that,

$$\begin{aligned} K\left(\frac{1}{\sigma_t}\right)^\eta &\geq K\left(\frac{1}{Q_\sigma}\right)^\eta \exp(U\eta t) && \text{using hypothesis 5.16,} \\ &\geq K'\eta'^t && \text{if } K \text{ and } \eta \text{ are large enough ,} \end{aligned}$$

with K' and η' some (fixed) parametrization of the exponential reevaluation scheme. Therefore, we obtain the same results as in Theorem 5.1. □

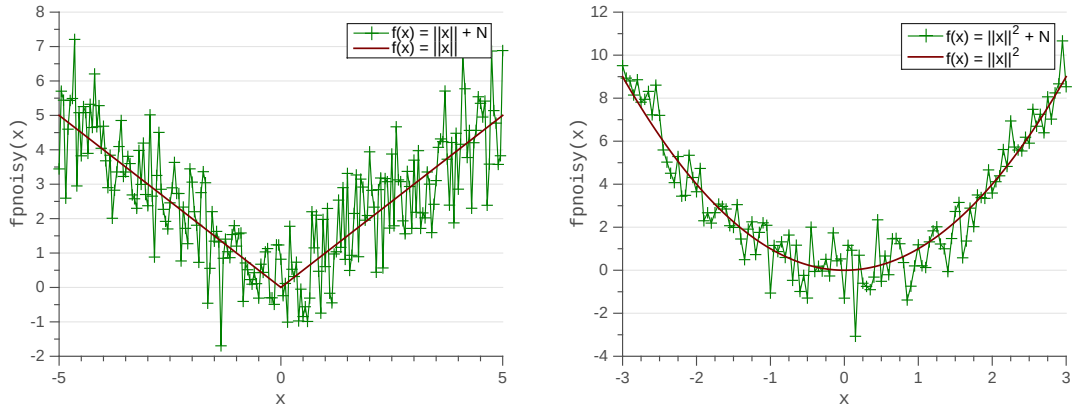
Corollary 5.2. *Let $n = n(t)$ be the number of evaluations at the end of iteration t. Then with probability at least $1 - \delta$*

$$\frac{\log(||x_n||)}{\log(n)} \to -\frac{R}{U\eta} \ . \tag{5.18}$$

Proof. Similar to the Corollary 5.1, we compute the amount of evaluations $n(t)$. We obtain $n(t) = K\left(\frac{1}{Q_\sigma}\right)^\eta \exp(U\eta)\frac{\exp(U\eta t)-1}{\exp(U\eta)-1}$ and use 5.15 to get the result. □

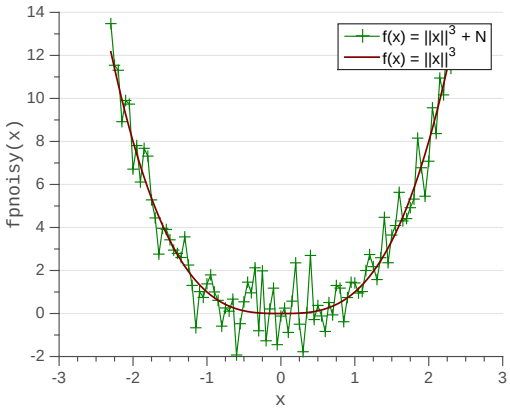
5.4 Polynomial: experimental work

We plot in figure 5.1 examples of the noisy fitness function for $p = 1, 2, 3$ and 4. We exhibit one simulation of the deterministic function in red and one simulation of the noisy version of the function in green. This shows the difficulty we may have to find the real optimum if we have access only to the noisy evaluations of the points.

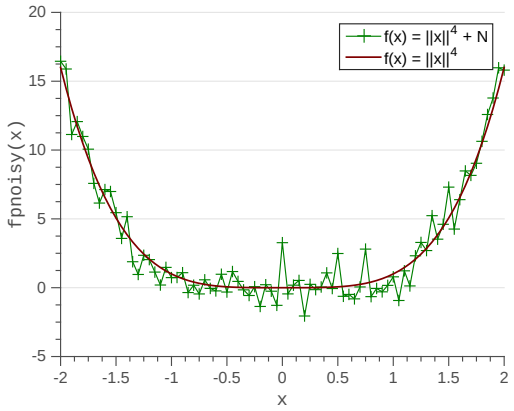


(a) p=1

(b) p=2



(c) p=3



(d) p=4

Figure 5.1: Fitness function with and without noise. Dimension $d = 1$

In this section we show experiments using a polynomial reevaluation. Algorithm 5.1 is the precise pseudo algorithm used for the experiments (based on [Auger *et al.*, 2011]) and the results are figure 5.2. We plot the logarithm of the number of evaluations in the x-axis and the logarithm of the Simple Regret in the y-axis. We observe approximate linear behaviour for the log-log scale. This is the same convergence ordered obtained theoretically for the exponential and adaptive schemes in previous sections.

Algorithm 5.1 (μ, λ)-σSA-ES algorithm with polynomial reevaluation $r_t = Kt^\eta$

Given d dimension, $\lambda \geq 5d$, $\mu \approx \lambda/4 \in \mathbb{N}$, $\tau \approx 1/\sqrt{d}$, $\tau_c \approx 1/n^{1/4}$
Initialize Parent $x_1 \in \mathcal{D}$, $\sigma_1 \in \mathbb{R}_+^d$, $t = 1$
while termination criterion not reached **do**
  for $j = 1, \ldots, \lambda$ **do**
    $\phi_j = \tau \mathcal{N}(0, 1)$
    $\Phi_j = \tau_c \mathcal{N}(0, \mathbb{I})$
    $z_j = \mathcal{N}(0, \mathbb{I})$
    Step-size $\sigma_{t,j} = \sigma \cdot \exp(\Phi_j) \exp(\phi_j)$ ▷ Mutation
    Offsprings $x_{t,j} = x_t + \sigma_{t,j} \cdot z_j$ ▷ Mutation
    Evaluate fitness r_t times and average $y_{t,j} = \frac{1}{r_t} \sum_{i=1}^{r_t} f(x_{t,j}, \omega_i)$ ▷ Evaluation
  end for
  Select from the μ best search points with largest fitness values $P_\lambda = \{(x_{t,j}, \sigma_j, y_j) : 1 \leq j \leq \lambda\}$ ▷ Selection
  $\sigma_t = \frac{1}{\mu} \sum_{\sigma_j \in P}$ ▷ Recombination
  $x_t = \frac{1}{\mu} \sum_{x_{t,j} \in P}$ ▷ Recombination
  $t = t + 1$
end while

We exhibit in table 5.2 a summary of the parameters of the experiments and the corresponding results. Each experiment includes 20 runs and we plot the average over the runs. To obtain the convergence rate (or slope) of the Simple Regret we only consider the latest evaluations. Therefore the aproximation of the convergence will represent a better estimate of the asymptotic behaviour. To choose μ and λ we use the recommendations on [Auger *et al.*, 2011]: $\lambda = 5d$ and $\mu = \lceil \lambda/4 \rceil$.

Type	K	η	p	dim	λ	μ	**s(SR)**
Experiment (a)	2	1	2	2	10	3	-0.2126
Experiment (b)	2	2	2	2	10	3	-0.3267
Experiment (c)	2	1	3	2	10	3	-0.1910
Experiment (d)	2	2	3	2	10	3	-0.3058
Experiment (e)	2	1	4	2	10	3	-0.1404
Experiment (f)	2	2	4	2	10	3	-0.2829

Table 5.2: Parameters and result of the experiments with polynomial reevaluation scheme

We observe that $s(SR)$ is usually smaller than $1/(2p)$, which is slightly better than the results of [Rolet and Teytaud, 2010b], but this effect may be due to a non-asymptotic effect. We also observe that the use of greater η turns out to imply better results. Not presented here are results with $\eta = 0$, with an unsurprising poor performance.

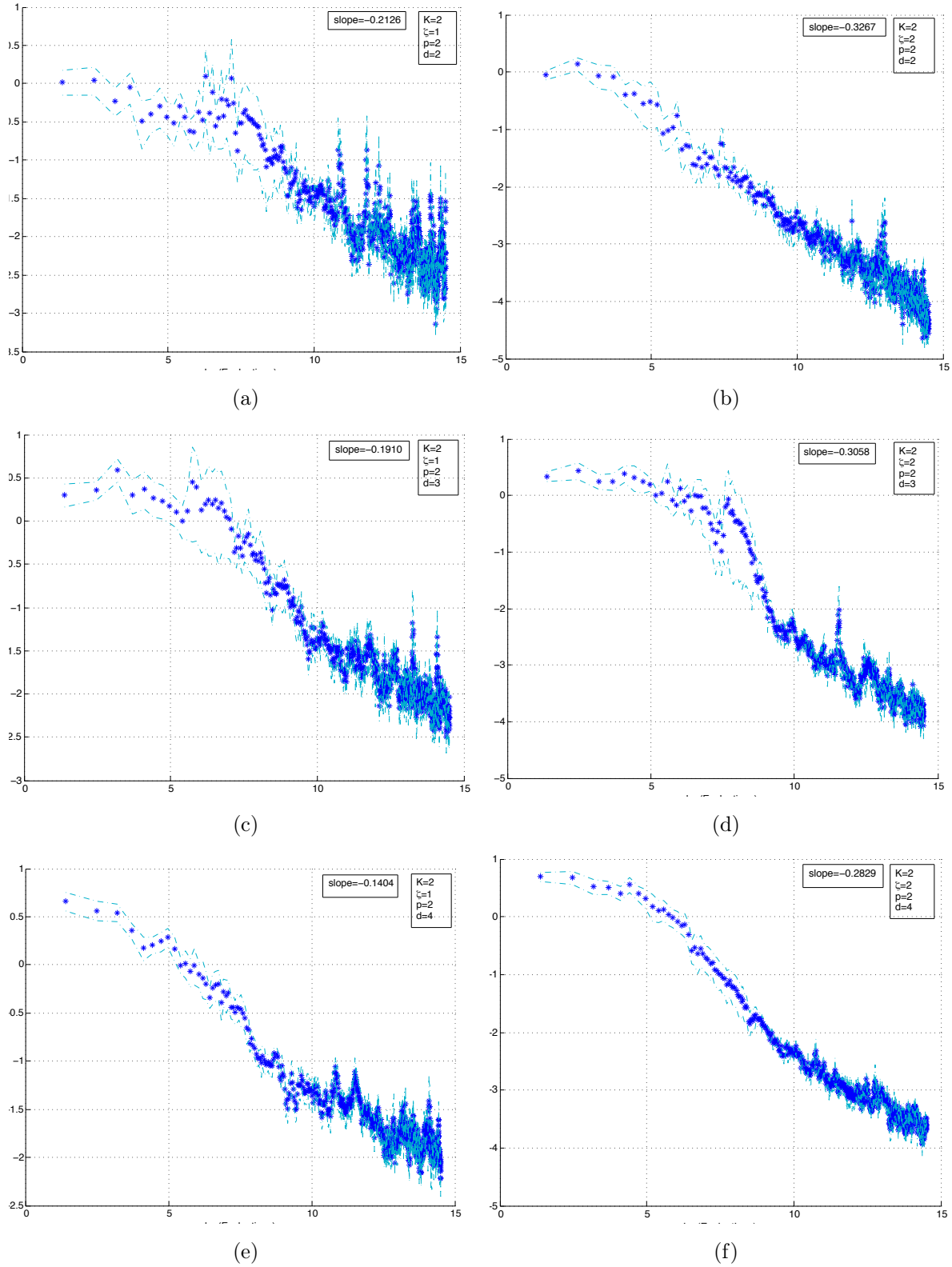


Figure 5.2: Results of experiments with polynomial reevaluation scheme. The x-axis represents $\log(n)$ and the y-axis represents $\log(SR_n)$. We observe a linear convergence on the log-log scale for the relationship between evaluations and Simple Regret.

5.5 Conclusion

We have studied the convergence behaviour of (μ, λ)-ES adapted to handle noise by reevaluation of the fitness function. We delivered a theoretical analysis of the order of convergence for the Simple Regret on two types of reevaluation: exponential and adaptive. In addition, an experimental analysis on other type of reevaluation: polynomial. The reevaluation schemes work depending on the iteration as exposed in Table 5.1.

The theoretical analysis yields a log-log order of convergence: the logarithm of the Simple Regret scales linearly with the logarithm of the evaluations. In the case of the exponential reevaluation scheme, we use the assumption of scale invariance to obtain the results. This assumption is popular in theoretical analysis of ES but it is not realistic.

We obtain also the same order of convergence for the adaptive reevaluation scheme. The adaptation uses the step-size to compute the number of evaluations necessary at each iteration.

The experimental analysis of the polynomial reevaluation scheme exhibit a similar result on the order of convergence as the theoretical analysis summarised above. We have the advantage to be able to observe also the convergence rate in this experiments. The polynomial scheme is convenient and it shows promising results experimentally, nonetheless we do not show theoretic results.

Chapter 6

Lower Bound Convergence Rate Evolution Strategies

We continue the work of Chapter 5 and determine now a lower bound for the convergence rate of Evolution Strategies. This lower bound is verified for the convergence of the Simple Regret, therefore we analyze the slope of Simple Regret: $s(SR)$.
We will present a general optimization algorithm, with processes of *recommendation* and *search*, but in a more general way than the algorithm presented in Chapter 4. We use here a well characterised *search* process to represent a simple Evolution Strategy. We also enumerate the necessary conditions to attain the bound presented, and comment on the cases where the conditions are met.
We obtain that $s(SR) > -1/2$ for simple Evolution Strategies that notably do not use search points far away from the optimum. The results are proved for a quadratic form, but they extend to any family that includes forms as the ones used here. We finish with an experimental verification of the theoretical result by comparing Evolution Strategies to a linesearch method for noisy optimization, found in [Shamir, 2013]. We observe that while the ES do not perform better than the bound showed theoretically on this work, the linesearch method achieves the optimal convergence rate for quadratic functions: $s(SR) = -1$.
The results presented are based on the conference publication [Astete-Morales *et al.*, 2015]. This is a joint work with Marie-Liesse Cauwet (TAO, INRIA Saclay) and Olivier Teytaud (TAO, INRIA Saclay, now Google).

6.1 Context

Evolutionary Algorithms are popular tools due to their wide applicability in optimization problems. In particular they show robustness in front of rugged fitness landscapes. This feature translates into a strong advantage of Evolutionary Algorithm to optimize functions corrupted by noise.
Evolution Strategies are a particular type of Evolutionary Algorithms used in continuous optimization. Their *mutation* operator creates an offspring by taking the parent of generation and adding some random perturbation to it. The random perturbation is usually Gaussian and controlled by a step-size. Therefore each offspring is produced from a random variable centered on the parent of the generation. This means that the mutation operator is more likely to create

offsprings "close" to the parent of the generation.

6.1.1 Typical convergence behaviour

The goal is to minimize the necessary number of queries to the oracle to find a good recommendation. We measure the convergence of the Simple Regret with regards to the number of function evaluations. As we saw in Chapter 5, the ES converge in a log-log order, but we did not ensure the convergence rate. We find results of the log-log convergence order in the work of [Arnold and Beyer, 2002; Coulom, 2012; Decock and Teytaud, 2013].

On the other hand, there are other algorithms that apparently have a better performance than the ES. We mention here linesearch method for noisy optimization, but we do not discard that there might be other type of algorithms also reaching optimal convergence rates (such as, algorithms that use models of the objective functions). In [Fabian, 1967] and [Shamir, 2013] we find two linesearch methods of zero order that use different ways to approximate the gradient of the function. They both obtain log-log convergence order for the Simple Regret, with convergence rate $s(SR) = -1$. More precisely, [Shamir, 2013] proves the convergence rate for strongly convex quadratic functions in a non-asymptotic way. And [Fabian, 1967] for a wider family of functions, asymptotically.

This work will in fact prove that there is a lower bound for the convergence rate of the ES. Therefore proving that they cannot reach the convergence rate of the optimal linesearch methods mentioned previously.

6.2 Preliminaries

Consider f a noisy function with additive noise as presented in Section 2.3. Noise is unbiased and with constant variance. The search points are x_n at evaluation n and the recommendations are $\hat{x}_n$. We denote the evaluations by $y_n = f(x_n, \omega)$. We denote also by $Z_n = ((x_0, y_0), \dots, (x_{n-1}, y_{n-1}))$ the first n pairs of search points and their respective noisy objective value.

We will study the convergence rate of the Simple Regret. In the proof, we will use a result from [Shamir, 2013] on the Cumulative Regret for noisy optimization algorithms (see Theorem 3.7 for details).

6.3 General Formalization of Algorithms

An optimization algorithm basically samples some search points, evaluates them and proposes a recommendation based on the information it has available. We can formalize a general optimization algorithm in Algorithm 6.1.

Algorithm 6.1 General Optimization Algorithm

Require: s random seed, p parameters, $\mathcal{I}$ initial internal state

1: $t = 0$
2: **while** not finished **do**
3: $\quad r = rand(s)$
4: $\quad \hat{x}_t = \texttt{OPT}(Z_n, r, t, p, \mathcal{I})$ ▷ Recommend
5: $\quad x_t = \texttt{SEARCH}(Z_n, r, t, p, \mathcal{I})$ ▷ Search Point
6: $\quad y_t = f(x_t, \omega)$ ▷ Evaluation
7: $\quad t = t + 1$
8: **end while**

The procedure `OPT` outputs a feasible point that stands as the approximation to the optimum. The procedure marked as `SEARCH` in Algorithm 6.1 outputs one or many search points that will be evaluated. Starting from $t = 1$, both procedures use the information from previous iterations.

The framework presented in Algorithm 6.1 is in fact very general. Thanks to the parameter r it includes possible randomized methods. Say we want to reproduce an algorithm that uses populations. Then a population of λ search points can be output by `SEARCH` by splitting the offspring reproduction in λ iterations, without varying the recommendation. In summary, Algorithm 6.1 encodes black box algorithms, including ES and linesearch methods such as Shamir and Fabian. Note that we make no assumption on the distance between the search points and the recommendations. Algorithms like Shamir and Fabian use in their advantage the search points as exploration, so it is even desirable that they are far away from recommendations. On the other hand, ES use `SEARCH` procedure that concentrates search points around the current recommendation.

6.3.1 Simple Evolution Strategy

In the context of Algorithm 6.1, we can specify the procedures to define an ES. In particular, we define the *search distribution* of the algorithm. Normally the sampling for the search points in ES is made around the current recommendation, therefore we define `SEARCH` as:

$$\texttt{SEARCH}(\zeta_t) = \texttt{OPT}(\zeta_t) + \sigma(\zeta_t) z(\zeta_t) \ . \tag{6.1}$$

Where $\zeta_t = (Z_t, r, t, p, \mathcal{I})$. Usually the stepsize $\sigma(\zeta_t)$ is updated at each iteration and sometimes for each coordinate. The random process $z(\zeta_t)$ is an independent d-dimensional random variable, with expectation:

$$\mathbb{E}(||z(\zeta_t)||^2) = d \ . \tag{6.2}$$

We also assume that the ES satisfies the following condition:

$$\exists C > 0 \text{ such that } \forall t \geq 0 \quad \mathbb{E}(\sigma(\zeta_t)^2) \leq C \mathbb{E}(||\hat{x}_t - x^*||^2) \ . \tag{6.3}$$

A *Simple Evolution Strategy* (Simple ES) will be an optimization algorithm that satisfies all the conditions mentioned above.

6.3.2 Evolution Strategies included in the formalization

We proceed to detail the ES that can be considered as Simple ES and the ones are discarded.
The assumption on Equation 6.2 is a light assumption. If it is not met, it can always be fixed by rescaling the stepsize $\sigma(\zeta_t)$ and afterwards satisfy the constraint.
The assumption on Equation 6.1 refers to the mutation process of an ES. The process described above matches the classical mutation on ES for continuous optimization. It is verified on ES with single parent or a μ/μ recombination, including weighted recombinations (for a complete overview on ES, see [Hansen *et al.*, 2015]).
The assumption on Equation 6.3 means that $\hat{x}_t$ and the stepsize $\sigma(\zeta_t)$ decrease at the same rate towards the optimum. It is verified in several situations according to literature:

- Scale invariant algorithm verify the property by definition. This assumption even if not realistic, it is used widely in theoretical analysis.
- The results on [Auger, 2005] on the sphere prove that $\sigma_t/||\hat{x}_t||$ converges to a distribution.
- Verification a posteriori on experimental results when algorithms converge [Beyer and Schwefel, 2002], in self-adaptive algorithms [Hansen and Ostermeier, 2001], and in most of Evolutionary Algorithms [Beyer, 2001].

Probably it is more useful to state which ES would not be covered among the Simple ES. Notably, algorithms that sample far away from the current estimate $\hat{x}_t$ in order to obtain or deduce more information on the function (as in surrogate models).

6.4 Theorem: Lower bound for Simple ES

We define the family $\mathcal{F}$ of all quadratic functions F that satisfy:

$$\begin{aligned} F : \mathcal{D} &\to \mathbb{R} \\ x &\mapsto \frac{1}{2}||x||^2 - (x \cdot x^*) \ , \end{aligned}$$

with $||x^*|| \leq 1/2$. We define its noisy counterpart in the Definition 6.1.

Definition 6.1. *The noisy counterpart of family F denoted by $\mathcal{F}_\mathcal{N}$ is defined as follows:*

$$\mathcal{F}_\mathcal{N} = \{f | f(x,\omega) = F(x) + \omega, F \in \mathcal{F}, Var(\omega) = 1\} \ .$$

Now we state the main result of this Chapter.

Theorem 6.1. *Let a simple Evolution Strategy as defined in Section 6.3.1 and assume that the simple ES solves $f \in \mathcal{F}_\mathcal{N}$. Then, for all $\alpha > \frac{1}{2}$,*

$$s(SR) > -\alpha \ .$$

Remark 6.1. *We assume here that we have a log-log convergence order and we only proof that if that is the case, then the convergence rate of the Simple Regret* can not *be lower than* $-1/2$.

Proof. By contradiction, let us assume that $SR_t \leq \frac{D}{t^\alpha}$ for some $\alpha > 1/2$ and $D > 0$. To show the contradiction we will use Theorem 3.7 that ensures that for a Simple ES there is at least one function $f \in \mathcal{F}$ for which $CR_t \geq 0.02 \min(1, d\sqrt{t})$. Let us compute CR_t of the Simple ES:

$$
\begin{aligned}
2CR_t &= 2\sum_{i=1}^{t} \left(\mathbb{E}f(x_i, \omega_i) - f(x^*)\right) && \text{by Def. of } CR \text{ Eq. 2.11 ,} \\
&= \sum_{i=1}^{t} \mathbb{E}\left(\|x_i\|^2 - 2(x^* \cdot x_i) + \|x^*\|^2\right) , \\
&= \sum_{i=1}^{t} \mathbb{E}\left(\|x_i - x^*\|^2\right) , \\
&= \sum_{i=1}^{t} \mathbb{E}\left(\|\hat{x}_i - x^* + \sigma_i \Phi_i\|^2\right) && \text{by Eq. 6.1 ,} \\
&\leq \sum_{i=1}^{t} \left(\mathbb{E}\|\hat{x}_i - x^*\|^2 + \mathbb{E}\sigma_i^2 \mathbb{E}\Phi_i^2\right) && \text{by independence ,} \\
&\leq \sum_{i=1}^{t} \left(\mathbb{E}\|\hat{x}_i - x^*\|^2 + d\mathbb{E}\sigma_i^2\right) && \text{by Eq. 6.2 ,} \\
&\leq 2(1 + dC) \sum_{i=1}^{t} \mathbb{E}(SR_i) && \text{by Eq. 6.3 ,} \\
\implies CR_t &\leq D(1 + dC)t^{1-\alpha} .
\end{aligned}
$$

Using Theorem 3.7 we have the contradiction. □

Let us note that Theorem 6.1 considers a particular type of family of quadratic functions, but the result applies for *any* family of functions that includes sphere functions (i.e. functions in $\mathcal{F}$) with additive noise.
With regards to the tightness of the result, it is not known whether ES reach $s(SR) = -1/2$. [Decock and Teytaud, 2013] prove that an ES with Bernstein Races with modified reevaluations depending on the fitness of the population can reach $s(SR) = -\alpha$ with α arbitrarily close to $-1/2$.

6.5 Experimental Verification

We present here experiments with three Algorithms: Shamir Algorithm, $(1+1)$-ES and UHCMAES ([Hansen *et al.*, 2009][1]). Shamir Algorithm reaches theoretically the convergence rate $s(SR) = -1$. The (1+1)-ES and UHCMAES are both contained in the set of algorithms concerned by the result in our Theorem of lower bound of the convergence rate for Simple Regret. We will see in this section that the experiments are consistent with the theoretical results.

[1] For the experiments we use the algorithm in `https://www.lri.fr/~hansen/` with option "uncertainty handle:on"

6.5.1 Fast convergence rate: Shamir Algorithm

The Shamir Algorithm, described in Algorithm 3.4, uses an estimation of the gradient to update its optimum at each iteration. The objective of this section is twofold: show how to write an algorithm in the form of the framework and show the experiments that exhibit the convergence rate of the algorithm. First we show that we can write an algorithm as the described framework in 6.1[2]. Note that the `SEARCH` defines the search points, using some random process, represented by r. Shamir Algorithm is proved to reach asymptotically $s(SR) = -1$ in the quadratic case (in [Shamir, 2013]). Notice also that this algorithm does not satisfy equation 6.1 nor 6.3 therefore it cannot be considered as a Simple ES.

Algorithm 6.2 SHAMIR ALGORITHM. Written in the general optimization framework. In this case $\mathcal{I}$ is a vector of t elements in the domain.

procedure $\mathsf{REC}(x_0, \ldots, x_{t-1}, y_0, \ldots, y_{n-1}, r, t, p, \mathcal{I})$
  if $||\mathcal{I}_t|| \geq B$ **then**
    $\mathcal{I}_t = B\frac{\mathcal{I}_t}{||\mathcal{I}_t||}$
  end if
  $\hat{x}_t = \frac{2}{t}\sum_{j=\lceil n/2 \rceil,\ldots,t}^{t} \mathcal{I}_j$
end procedure
procedure $\mathsf{SP}(x_0, \ldots, x_{t-1}, y_0, \ldots, y_{t-1}, r, t, p, \mathcal{I})$
  if $n = 0$ **then**
    $\mathcal{I} = (0)$
    Return $x_0 = 0$
  end if
  Compute $x_t = x_{t-1} + \frac{\varepsilon}{\sqrt{d}} r$
  Compute $\hat{g} = \frac{\sqrt{d} y_{t-1}}{\varepsilon} r$
  Compute $\mathcal{I} = (\mathcal{I}, x_{t-1} - \frac{1}{\lambda t}\hat{g})$
end procedure

Input: $p = (\lambda, \varepsilon, B) \in \mathbb{R}_+ \times (0, 1] \times \mathbb{R}_+$, $s =$ random seed
$t \leftarrow 0$
loop
  Generate $r \in \{-1, 1\}^d$, uniformly and randomly
  $\hat{x}_t = \mathsf{REC}(x_0, \ldots, x_{t-1}, y_0, \ldots, y_{t-1}, r, t, p, \mathcal{I})$
  $x_t = \mathsf{SP}(x_0, \ldots, x_{t-1}, y_0, \ldots, y_{t-1}, r, t, p, \mathcal{I})$
  $y_t = f(x_t, \omega)$
  $t \leftarrow t + 1$
end loop

[2]The reader can compare the original Algorithm 3.4 with the framework presented here. Even though Algorithm 6.2 may seem more complicated, the advantage is that using the framework we have exact information on the different processes involved in the optimization

Results in figure 6.1 correspond to the mean of 21 runs of Shamir Algorithm for each dimension tested. Note that we plot directly the slope (or the convergence rate) of the Simple Regret. The function is the noisy sphere with optimum $x^* = 0.5$ and Gaussian noise. We observe that $s(SR)$ is always lower than $-1/2$, converging to -1. Always better than the lower bound we have proved in Theorem 6.1.

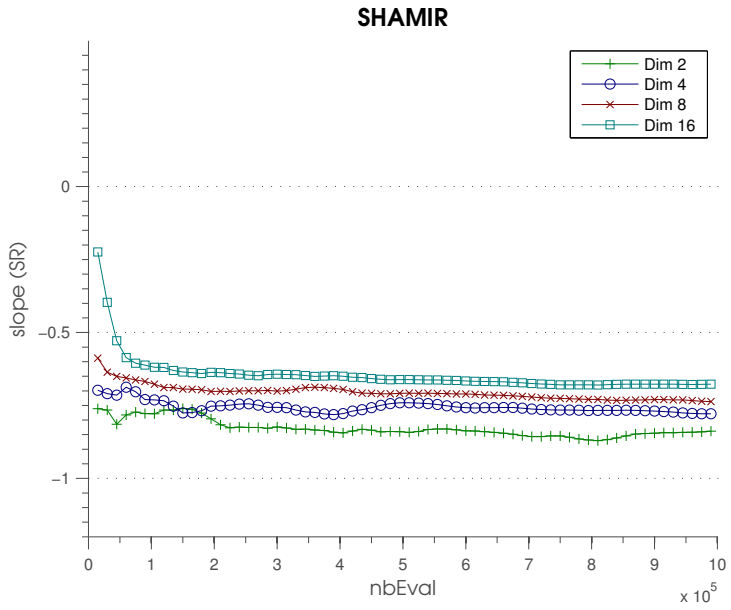


Figure 6.1: Shamir Algorithm on the noisy sphere function. Average over 21 runs of the algorithm. The maximum standard deviation for all averages presented here is 10^{-3}

6.5.2 Slow convergence rate: UHCMA and $(1+1)$- ES

We test experimentally two ES. First, the UHCMAES (from [Hansen *et al.*, 2009]) that corresponds to CMA with an uncertainty handling tool to solve noisy objective functions. And second, the $(1+1)$- ES with reevaluation (see Chapter 5 for more details on the reevaluation schemes). Both algorithms fall in the category of Simple ES.
Results in figure (6.2) correspond to the mean of 21 runs of UHCMA Algorithm for each dimension tested. The function is the noisy sphere $f(x, w) = F(x) + 0.3\omega$ where $\omega \sim \mathcal{N}(0, 1)$ with optimum $x^* = 0.5$.

Results in figure 6.3 correspond to the mean of 400 runs of $(1 + 1)$-ES Algorithm for each dimension tested. The function is the noisy sphere with optimum $x^* = 0.5$ and Gaussian noise. We observe that $s(SR)$ is never under $-1/2$, as predicted in Theorem 6.1.

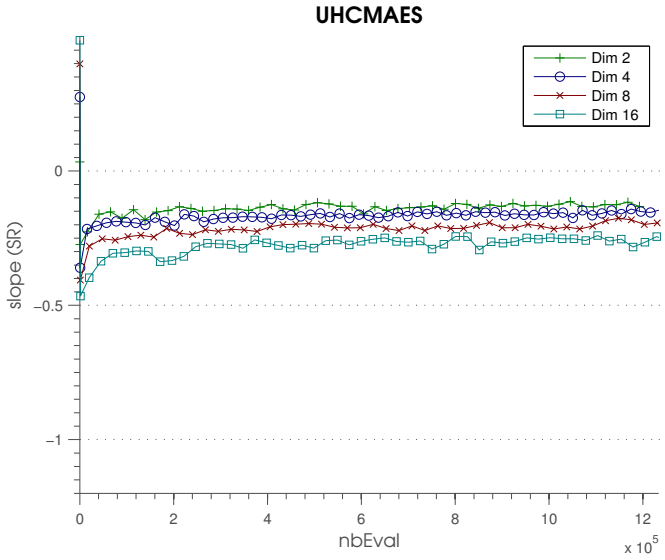


Figure 6.2: UHCMAES Algorithm on the noisy sphere function. Average over 21 runs of the algorithm. The maximum std for all averages presented here is 1

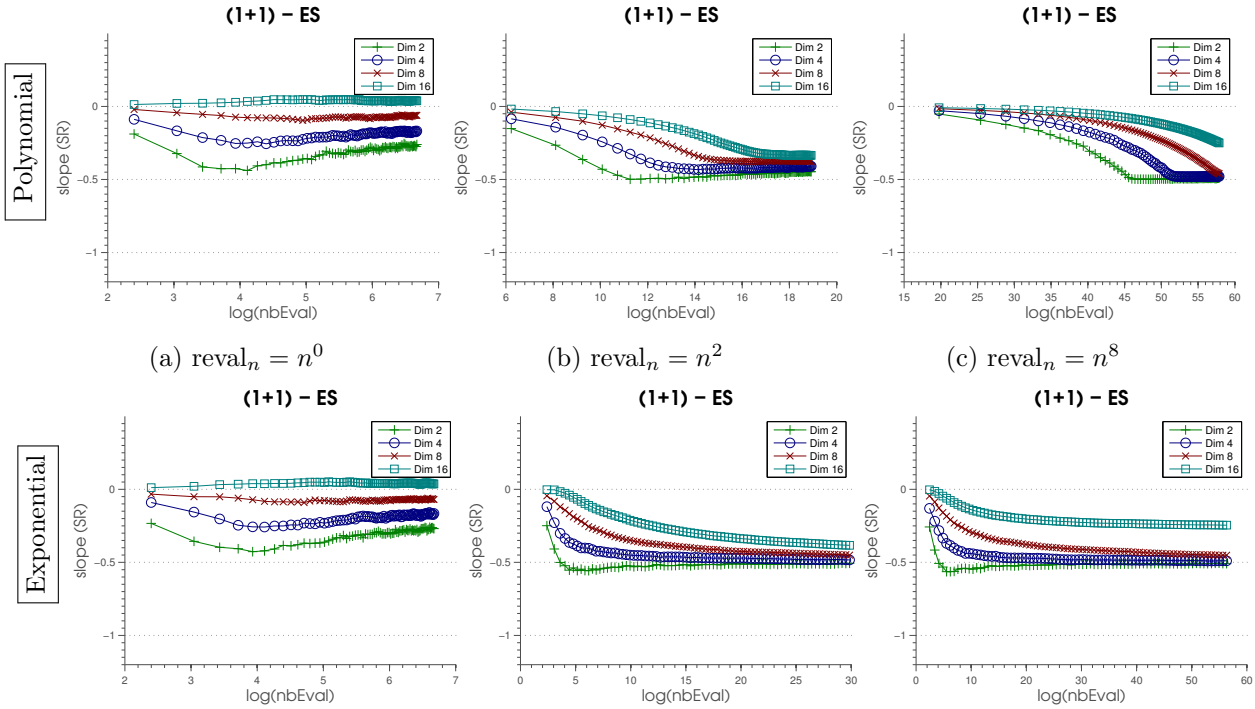
6.6 Conclusion

We have shown with Theorem 6.1 that Evolution Strategies, at least under their most common form, cannot reach the same rate as other noisy optimization algorithms, for instance the linesearch algorithms in [Shamir, 2013; Fabian, 1967]. The significant difference between these algorithms seems to be the `SEARCH` process. For the ES, the `SEARCH` process favors the points *close* to the current recommendation, while the linesearch algorithms do not satisfy this property.

The linesearch algorithms, Shamir and Fabian, both exhibit $s(SR) = -1$, non-asymptotically and asymptotically respectively. While we prove here that the ES can only reach $s(SR) = -1/2$ at most. This answers the conjecture raised in [Shamir, 2013]. But, we should also point out that there is no proof that linesearch algorithms are the only ones able to reach optimal convergence rates.

We notice also that Theorem 6.1 covers many pattern search methods. The only requierement is the form of the `SEARCH` process (Section 6.3.1), which we denote by *simple Evolution Strategy*. Nonetheless, the results do not cover Evolution Strategies with large mutations. This could be a good way to speed up the convergence of an Evolution Strategies.

The experiments are done for the sphere function with additive noise. We contrast the result of "fast" linesearch methods and "slow" ES. We use for the linesearch the Shamir Algorithm (see Alg. 3.4 or [Shamir, 2013] for more information). For the ES we use the UHCMAES and $(1+1)$-ES. We confirm the theoretical results.



(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 6.3: $(1 + 1)$-ES Algorithm on the noisy sphere function. Average over 400 runs of the algorithm. The maximum std for all averages presented here is 0.025. First row of plots presents Polynomial reevaluation and the second row Exponential reevaluation.

Chapter 7

Approximations of Simple Regret: Inconsistent Performance

This chapter drifts apart from the analysis of specific problems/algorithms. Instead, we focus on the *way* to measure the performance of algorithms. We present the consequences of using approximated performances measures to measure the performance of noisy optimization algorithms. The performance measure involves the "quality" of the recommendation output by the algorithm. We can use the "Regret" to measure the quality. The Regret accounts for the loss of choosing the recommendation output by the algorithm instead of the real optimum. In other words, the regret is the difference, over the codomain, of the recommendation and the optimum.

The most natural way to measure the quality of the recommendation is to use the Simple Regret. It is defined as the difference between the recommendation at time t and the real optimum on the image space. But some optimization algorithms also produce search points, where the function will be evaluated in order to explore the domain. The search points can be exactly the recommendations, but not necessarily. A particular problem appears when we cannot distinguish the recommendation and search points. In that case we can imagine that the performance can be measured by looking at difference between the search points at time t and the real optimum on the image space. We will see that the use of this performance measure, which we will call Approximate Simple Regret, can be misleading and return contradictory results.

Even if we have access to the separation between recommendation and search points, noisy optimization problems present another problem to evaluate the performance of an algorithm in practice. Since we obtain noisy evaluations of points, we cannot ensure to have chosen the best point to be recommended at iteration t. Therefore, the Simple Regret at time t can be larger (and therefore worse) that the Simple Regret at time $t+1$. In that case we cannot evaluate the algorithms with a stop criterion of first hitting time. We will see that another approximation, called Robust Simple Regret, non-increasing and using recommendation points cannot solve the contradictions.

We analyze the use of Simple Regret, Robust Simple Regret and Approximate Simple Regret. The two latter aim to emulate the behaviour of Simple Regret. We show that they lead to incompatible performance evaluations of the same algorithms over the same class of functions. This will be shown by analysing the convergence of each Regret using the noisy sphere function and specific algorithms.

The results presented are based on the conference publication [Astete-Morales *et al.*, 2016b]. This is a joint work with Marie-Liesse Cauwet (TAO, INRIA Saclay) and Olivier Teytaud (TAO, INRIA

Saclay, now Google).

7.1 Context

To evaluate the use of the regrets to measure the performance of algorithms, we will focus on one simple continuous noisy problem: the sphere function F with additive noise. The noisy function f that is what the algorithm see is defined by:

$$f : \mathbb{R}^d \times \Omega \to \mathbb{R}$$
$$(x, \omega) \mapsto F(x) + \omega = ||x - x^*||^2 + \omega \ ,$$

where $\omega \sim \mathcal{N}(0, 1)$. The objective of the noisy optimization algorithms is to find $\hat{x}$ such that $\mathbb{E}_\omega f(\hat{x}, \omega)$ is approximately minimum. Note that the expectation is only over ω, it does not include the possible randomization of the algorithm.

7.1.1 Simple Regret and its approximations

The Simple Regret (SR) is defined at time t by

$$SR_t = \mathbb{E}_\omega \left[f(\hat{x}_t, \omega) - f(x^*, \omega) \right] = F(\hat{x}_t) - F(x^*) \ .$$

The SR is a way to measure the *precision* of the algorithm, by verifying that the image of the recommendation $\hat{x}_t$ is close to the image of the real optimum x^*. The evaluations of f that the algorithm uses to output a recommendation are noisy. Therefore, it is not sure that the sequence SR_t is non-increasing. The algorithm may output a recommendation in time t that is much better than the one of the $t+1$ and drop it because of the noisy evaluation of the good solution said that it was a bad one. The first hitting time (FHT) criterion cannot be used with the SR.

The Approximate Simple Regret (ASR) also aims to measure the performance of the algorithm but using only the evaluated points of the domain. That is is, it uses the search points. But taking into consideration the best search points with regards to the distance to the optimum on the codomain. ASR is by definition non-increasing.

$$ASR_t = \min_{\tau \leq t} F(x_\tau) - F(x^*) \ . \tag{7.1}$$

We also discuss the Robust Simple Regret (RSR), which incorporates recommendation points and it evaluates how good are they over several iterations of the algorithm.

$$RSR_t = \min_{t' \leq t} \left(\max_{t' - g(t') < \tau < t'} F(\hat{x}_\tau) - F(x^*) \right) \ . \tag{7.2}$$

Essentially the RSR will acknowledge the best SR since the start of the run, and that is also maintained as the best during $g(t')$ consecutive instants. We will assume that g is a polylogarithmic function. That implies that g is "large enough" so that the recommendation is actually being the best for several instants but not so large that we have to have t too big before acknowledge a good recommendation. The definition of RSR suggests that it will outplay ASR for two reasons. The

first is that it uses the recommendations output by the algorithm. The second is that it checks that the recommendation is good over several iterations. In addition it is also non-increasing. We will see that even these advantages are not enough to make it a satisfactory approximation of Simple Regret. To measure the convergence rate of algorithms we will use the slope of each regret.

7.2 Random Search Heuristics: ASR overestimates SR

We include here two algorithms: Random Search and Evolution Strategies. For the latter we consider both Evolution Strategies without and with reevaluation. Check Chapter 5 for details on the modification of Evolution Strategies to handle noisy evaluations.

7.2.1 Random Search

We consider Random Search as in Algorithm 3.5. We consider that the search distribution of Random Search works as follows. Each search point is selected once and only once from a uniform distribution over $\mathcal{D} = [0,1]^d$. We assume the optimum x^* belongs to $\mathcal{D}$. The recommendation point is the best search point so far.

Property 7.1. *$s(ASR) = O(-2/d)$ a.s. for Random Search on the noisy sphere function.*

Proof. From the work in [Deheuvels, 1983] we know that the point that is closest to the optimum x^* is almost surely at distance $O(t^{-1/d})$ within a logarithmic factor, considering a sample of points of size t. Hence we obtain the result using the definition of ASR. □

We provide in the Appendix, Section 10.3, experiments that suggest that the Random Search in fact converges for the noisy sphere. Even more, according to the experiments the upper bound might be reached.

In the case of SR, we prove that $s(SR)$ is not negative. Therefore, the convergence measured by ASR overestimates the real convergence measured by SR.

Theorem 7.1. *For all $\beta > 0$, $\mathbb{E}[SR_t] \notin O(t^{-\beta})$ for Random Search on the noisy sphere function.*

For the formal proof we refer the reader to the Appendix, Section 10.4. Roughly speaking, the argument is that with probability non-zero, we select a recommendation point that is not the one with the best fitness.

7.2.2 Evolution Strategies

The work from [Arnold and Beyer, 2006] shows that an ES without any adaptation to noise stagnates around some step-size and at some distance from the optimum. We can also find experimental results for ES solving noisy functions in [Beyer, 1998]: they stagnate at some step-size and some distance to the optimum. These divergence results suggest that the ES act only as a more sophisticated version of Random Search for the optimization of noisy functions. We propose a conjecture on the convergence rates for ES without a noise treatment.

Conjecture 7.1. *The convergence rate for regrets of Evolution Strategies are equal to the convergence rates for Random Search on the noisy sphere function.*

Now we consider an ES with noise treatment: reevaluation of search points. We will denote it by ES-r, where r represents the reevaluation scheme. We know by Chapter 5 that ES-r converge for the SR if they have the property of step size scaling as the distance to the optimum. We also know from the work on Chapter 6 that they cannot be faster than $s(SR) = -1/2$. And from experiments, we can see that they seem to reach $s(SR) = -1/2$. For more details on the convergence of ES-r and experiments go to Chapter 5. For more details on the lower bound for the convergence of ES-r go to Chapter 6. Therefore we propose:

Conjecture 7.2. $s(SR) = -1/2$ *with probability* $1 - \delta$ *for ES-r on the noisy sphere function.*

The conjecture 7.2 is valid for some ES that satisfy the conditions on detailed on Chapter 5. In other words, ES must have a reevaluation strategy, represented by r. The strategy can be either exponential or polynomial reevaluation. And also the algorithm must satisfy the condition over the step-size scaling as the distance to the optimum (see condition 5.8). Using this conjecture, we can prove that $s(ASR)$ is strictly better. We modify the algorithm ES-r (Algorithm 7.1)and to obtain $s(ASR) = -1/2 - 2/d$. Let us present the modification of the algorithm, called MES-r in Algorithm 7.2. We will modify the stage of generation by producing additional offsprings. And we modify the stage of evaluation by evaluating these additional offsprings. Nonetheless, the additional offsprings will not be taken into consideration for creating the recommended point.

Theorem 7.2. *Let* $0 < \delta < 1$. *Assume that*

$$||\hat{x}_n|| = \Theta(\sigma_n) \ , \tag{7.3}$$

$$\frac{\log(||\hat{x}_n||)}{\log(n)} \xrightarrow{n\to\infty} -\frac{1}{2} \text{ with probability } 1 - \delta \ . \tag{7.4}$$

Then $s(ASR) = -1/2 - 2/d$ *with probability* $1 - \delta$ *for MES-r on the noisy sphere function.*

Remark 7.1. *Just as in Conjecture 7.2, we need to assume the step-size scaling as the distance to the optimum. Note that we fix the reevaluation strategy as exponential, on Algorithm 7.2. We assume also in the proof of Theorem 7.2 that Conjecture 7.2 is true.*

Algorithm 7.1 ES-r

Input: μ, σ and r
Initialize: Parent $\hat{x}$, stepsize σ
Initialize: $i \leftarrow 1$
while not terminate **do**
  for $j \in \{1, \ldots, \lambda\}$ **do**
    Mutation: $x_j \leftarrow \hat{x} + \sigma\mathcal{N}$
    Eval.: $y_j \leftarrow \frac{1}{r(i)} \sum_{j=1}^{r(i)} f(x_i, \omega_i)$
  end for
  Selection: select μ best out of $(x_i)_1^\lambda$
  Update $\hat{x}$ using μ offsprings and σ
  $i \leftarrow i + 1$
end while
return $\hat{x}$

Algorithm 7.2 MES-r

Input: μ, σ and r
Initialize: Parent $\hat{x}$, stepsize σ
Initialize: $i \leftarrow 1$
while not terminate **do**
  for $j \in \{1, \ldots, \lambda\}$ **do**
    Mutation: $x_j \leftarrow \hat{x} + \sigma\mathcal{N}$
    Eval.: $y_j \leftarrow \frac{1}{r(i)} \sum_{j=1}^{r(i)} f(x_i, \omega_i)$
  end for
  Selection: select μ best out of $(x_i)_1^\lambda$
  Update $\hat{x}$ using μ offsprings and σ
  for $j \in \{1, \ldots, r(i)\}$ **do**
    Mutation: $x'_j \leftarrow \hat{x} + \sigma\mathcal{N}$
    Evaluation: $y'_j \leftarrow f(x'_i, \omega_i)$
  end for
  $i \leftarrow i + 1$
end while
return $\hat{x}$

Proof. We have $r_n = K\eta^n$. In this proof we will index the recommendation and search points by the number of *iterations* instead of the number of evaluations. For ES-r, the recommendation point of the iteration n is the corresponding center of the offspring distribution $\tilde{x}_n$. The step-size σ_n corresponds to the standard deviation of the offspring distribution. The search points of iteration n are the λ offsprings produced in the iteration, denoted $\{x_n^{(i)} : i = 1, \ldots, \lambda\}$.

We define e_n the number of evaluations until the iteration n. From Algorithm 7.2 we have $e_n = \lambda \sum_{i=1}^{n} r_i$ for an ES with reevaluation r. By hypothesis (Equation 7.4) we know the convergence of the sequence S_n defined as the logarithm of the recommendation points, indexed by the number evaluations, divided by the logarithm the number of evaluations. Therefore, the sequence $\mathcal{S}_n = \log(||x_n||)/\log(e_n)$ is a subsequence of S_n, hence convergent to the same limit, with the same probability. The relation in Equation 7.3 also remains the same after the index modification. From these facts we can immediately conclude that $\exists \quad n_0$ such that

$$\sigma_n = \Theta(e_n^{-1/4}) \text{ for } n \geq n_0. \tag{7.5}$$

For the MES-r algorithm, the ASR_n is defined as:

$$ASR_n = \min_{m \leq n} \min_{1 \leq i \leq \lambda + r_m} \|x'^{(i)}_m\|^2 \ , \tag{7.6}$$

where $\{x'^{(i)}_n : i = 1, \ldots, \lambda + r_m\}$ considers all the search points at iteration n. Both the "real" and the "fake" ones. We will find a bound for ASR_n, which will lead us to the convergence rate of the ASR for the MES-r algorithm.

Let p_x be the probability density at point x of the offsprings in iteration n. Therefore it is the probability density of a Gaussian centered at $\tilde{x}_n$ and with variance σ_n^2. At the origin:

$$\begin{aligned} p_0 &= \frac{1}{(2\pi)^{d/2}\sigma_n^d} \exp\left\{-\frac{1}{2}\frac{(-\tilde{x}_n)^T(-\tilde{x}_n)}{\sigma_n^2}\right\} , \\ &= \Theta(\sigma_n^{-d}) , \\ &= \Theta(e_n^{d/4}) . \end{aligned} \tag{7.7}$$

Now, at iteration n, we can bound the probability to have at least one offspring with norm less than $\varepsilon > 0$

$$\begin{aligned} \mathbb{P}(\exists i : \|x'^{(i)}_n\| \leq \varepsilon) &\leq (\lambda + r_n)\mathbb{P}(\|x'^{(i)}_n\| \leq \varepsilon) , \\ &\leq (\lambda + r_n) \textstyle\int_{\|x\|\leq\varepsilon} dp_x . \end{aligned}$$

By Equation 7.7,

$$\mathbb{P}(\exists i | \|x'^{(i)}_n\| \leq \varepsilon) = \Theta(r_n \cdot e_n^{d/4}\varepsilon^d) = \Theta(1) ,$$

if $\varepsilon = \Theta(e_n^{-1/4} \cdot r_n^{-1/d})$. Therefore we obtain:

$$\begin{aligned} ASR_n &\leq \min_{1\leq i\leq\lambda+r_n} \|x'^{(i)}_n\|^2 , \\ &\leq \varepsilon^2 , \\ &= O(e_n^{-1/2} \cdot r_n^{-2/d}) , \\ &= O(e_n^{-1/2-2/d}) . \end{aligned}$$

Since $e_n = (1+\lambda)\sum_{i=1}^n r_i = (1+\lambda)\cdot R\sum_{i=1}^n \zeta^i = \Theta(r_n)$, we have the result:

$$s(ASR) \leq -1/2 - 2/d .$$

□

7.3 Noisy Linesearch Algorithms: ASR underestimates SR

We analyze two algorithms that use estimated gradient to find the optimum of noisy functions, both presented in Section 3.2.3. The first one is Shamir that uses a one point technique to estimate the gradient. The other one is Fabian, that uses finite differences to estimate the gradient. They have both proved to be optimal for SR. $s(SR) = -1$ in expectation for Shamir over quadratic functions and $s(SR) = -1$ approximately and asymptotically for limit values of parameters for Fabian for smoothly enough functions.

7.3.1 Shamir

The following result is a direct consequence from a wider result for more general functions proved in [Shamir, 2013].

Theorem 7.3. *$s(SR) = -1$ in expectation for Shamir algorithm on the noisy sphere function.*

The work on [Shamir, 2013] proves results in expectation. But if we assume the same results occur for almost sure convergence, then we obtain immediately that the ASR underestimates the convergence of the Shamir algorithm.

Theorem 7.4. *If Theorem 7.3 is also valid a.s. then $s(ASR) = 0$ for Shamir on noisy sphere function.*

Proof. Assume that the recommendations of Shamir $\hat{x}_i$ converge a.s. to the optimum x^*. By definition of the Shamir algorithm, the sequence of search points is at a constant distance η from x^*. Therefore, $\min_{j\in\{1,\dots,i\}} ||x_j - x^*||$ is lower bounded by η. Which implies that $s(ASR) = 0$ almost surely. □

Note that the consequences of Theorem 7.4 apply as long as the problem has a unique optimum, the sequences of recommendations of the algorithm converge a.s. to the optimum and the sequence of search points is such that the distance between the search points and the optimum is always constant. Therefore the result is wider and not only for the Shamir algorithm. Even so, the result for Shamir algorithm is only valid if we assume the conjecture that the results in Shamir's work are also valid for almost sure convergence.

7.3.2 Fabian

The following result in Theorem 7.5 comes from the results proved in [Fabian, 1967].

Theorem 7.5. *Let s be an even positive integer and F be a function $(s+1)$-times differentiable in the neighborhood of its optimum x^*. Assume that its Hessian and its $(s+1)^{th}$ derivative are bounded in norm. Assume that the parameters given in input of Algorithm 3.3 satisfy: $a > 0$, $c > 0$, $\alpha = 1$, $0 < \gamma < 1/2$ and $2\lambda_0 a > \beta_0$ where λ_0 is the smallest eigenvalue of the Hessian. Let $\beta_0 = \min(2s\gamma, 1 - 2\gamma)$. Then, a.s.:*

$$n^{\beta}(\tilde{x}_n - x^*) \to 0 \ \forall\ \beta < \beta_0/2\ . \tag{7.8}$$

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

We prove here the convergence rate for ASR in the form of the following theorem:

Theorem 7.6. $s(ASR) = -\min(2\beta, 2\gamma)$.

The detail of the proof is in Appendix 10.5. We can deduce directly the $s(ASR)$ using the parameters for optimal convergence rate of SR, with the following Corollary:

Corollary 7.1. *$s(ASR) \to 0$ when $\gamma \to 0$.*

In Corollary 7.1 the fact that γ tends to 0 is required to obtain the result in 7.5. Therefore, we obtain that Fabian proves the algorithm is optimal for SR. Simultaneously, we prove that the algorithm does not converge for ASR using the same parametrization that yields optimal convergence for SR.

7.3.3 Adapt algorithms

We can adapt both Shamir and Fabian so that they evaluate the recommendation points and we will obtain $s(ASR) = -1$. All we need to do is evaluate also the recommendation points. Even if they are not used anywhere in the algorithm. This is a serious problem: adding a senseless feature such as revaluate points and not use this information help us *improve* the results in terms of approximating the SR. But the objective of the algorithm is to *reduce* the amount of function evaluations.
On the contrary, recall that for the analysis of MES-r we add extra evaluations and we obtain a better convergence rate for ASR than for SR.

7.4 General Results for SR, ASR and RSR

The issue treated in this chapter is the gap between the regrets. Ideally we should have some convergence of the ASR and RSR to SR in some way. Nonetheless, this is not the case. We have seen over the sections 7.2 and 7.3 that ASR both underestimates and overestimates the performance of algorithms, depending on the type of algorithm. In the case of Evolution Strategies without a noise treatment, they converge for ASR. In the case of SR, ES do not converge. On the contrary, for algorithms in using estimated gradients, ASR does not report that they converge. When in fact, the convergence is optimal for SR.
For the RSR, we have that by definition $s(RSR) \leq s(SR)$. Therefore, $s(RSR)$ is a correct lower bound for $s(SR)$. Nonetheless, it is not a tight bound. We show that by modifying slightly the algorithm, we obtain that $s(RSR) \leq s(ASR)$ without reporting any change on $s(SR)$. Let algorithm A with search points $(x_t)_{t\geq 1}$ and algorithm A^g with search points $(x_t^g)_{t\geq 1}$. The search points of A^g are repeated search points of A. The definition uses a polylogarithmic function g depending of the iteration t as follows:

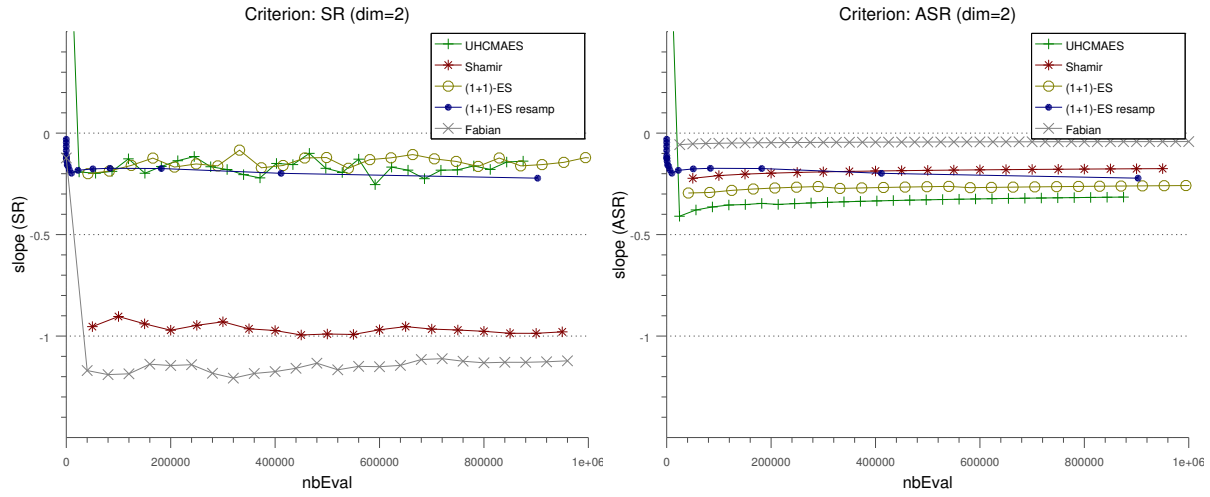
$$
\begin{array}{lll}
x_1^g = x_1 \ , & x_{g(1)+2}^g = x_2 \ , & \cdots \quad x_{i+\sum j=1^{i-1} g(j)}^g = x_i \ , \\
x_2^g = x_1 \ , & x_{g(1)+3}^g = x_2 \ , & \\
\vdots & \vdots & \vdots \\
x_{g(1)+1}^g = x_1 \ , & x_{g(2)+1}^g = x_2 \ , & \cdots \quad x_{i+\sum j=1^{i} g(j)}^g = x_i \ .
\end{array}
$$

The recommendation points of A^g are defined as $\hat{x}_i^g := x_i^g$ for any i.

By the definition of the regrets we do not have more information about the relationship among them. We will see them in action over five algorithms, over the same noisy optimization problem.

7.5 Experiments

We present experimental results for part of the algorithms theoretically analyzed[1]. We will analyze the convergence rate of these algorithms. We will plot the convergence rate (or slope) of the Simple Regret for each number of function evaluations.
As in the theoretical part of this study, the function to optimize is the noisy sphere: $f(x) = \|x - x^*\|^2 + \vartheta\mathcal{N}$ where $\vartheta = 0.3$ and $\mathcal{N}$ is a standard Gaussian distribution[2]. The dimension of the problem is $d = 2$. The results in Figure 7.1 correspond to the mean of 10 runs for each of the algorithms.



(a) Simple Regret

(b) Approximate Simple Regret

Figure 7.1: Figure 7.1a presents the Slope of Simple Regret for each algorithm on the first $(1 \cdot 10^6)$ function evaluations. Stochastic Gradient algorithms reach $s(SR) = -1$ while the evolutionary algorithms present $s(SR) = -0.2$. Figure 7.1b presents the Slope of Approximate Simple Regret. Observe that the performance of the algorithms is inverted with regards to the figure 7.1a : now the Stochastic Gradient algorithms have the worse performance.

[1] In addition, the experimental results we include the algorithm $UHCMAES$, as another example of an ES. For more information, see [Hansen *et al.*, 2009].

[2] The choice $\vartheta = 0.3$ is made only to illustrate in the experiments the effect of the regret choice in a reasonable time budget. The noise is weaker than in the case of a standard Gaussian and the algorithms can deal with it faster. The optimum x^* for the experiments of each algorithm is different, which does not affect the result since the regret compares the function value on the search/recommended points and on the optimum.

Algorithms	Set of parameters
UHCMAES [Hansen *et al.*, 2009]	$x_{\text{initial}} = 1$, $\sigma_{\text{in}} = 1$
Shamir	$\varepsilon = 0.3$, $\lambda = 0.1$, $B = 3$
$(1+1)$-ES	
$(1+1)$-ES-r	reevaluation= 2^n
Fabian	s=4 $\alpha = 1$, $\gamma = 0.01$

The results in figure 7.1a show the convergence rate for the different algorithms with regards to the SR. The budget is limited to $(1 \cdot 10^6)$ function evaluations. We can see clearly the difference between linesearch algorithms, Fabian and Shamir and ES. The algorithms Fabian and Shamir achieve $s(SR) = -1$ whereas the ES presented cannot do better than $s(SR) = -0.25$. The figure 7.1b shows that the use of ASR changes completely the performance of the algorithms. In this case, the gradient-based algorithms are the ones with the worst performance. The results support the theoretical work (and the conjectures) presented in sections 7.2 and 7.3.

7.6 Conclusion

We analyze the use of approximations of Simple Regret and how they give inconsistent results at measuring the performance of algorithms. For ASR we obtain that it underestimates the performance of algorithms that estimate the gradient. This can be solved by modifying the algorithm and evaluating recommendation points so that they can be taken into account for ASR. For the case of EAs, ASR overestimates their performance. There is no easy way to solve this issue by modifying the algorithms.

RSR on the other hand solves partly the issues. It is by definition a valid lower bound for the performance in terms of SR for any algorithm. Unfortunately, this bound is not tight. We prove that a slight modification of the algorithm implies that RSR is also a lower bound for ASR, without implying any modification on SR.

Even though SR is the natural way to measure the precision of the recommendation given by the algorithm, its not necessarily non-increasing nature does not allow it to be used easily in practice for the performance evaluation of evolutionary algorithms.

We have compared convergence rates with different mode of convergence. There is room for refinement of the results.

Chapter 8

Runtime Analysis of Optimization Algorithms over Discrete Noisy Functions

In this chapter we explore the runtime of algorithm for discrete noisy optimization. We inspire from the work presented on the previous chapters and use a reevaluation scheme to adapt a regular algorithm for discrete optimization, in order for it to be able to handle the noisy version of the discrete problem.
We obtain theoretical results for the runtime of the modified algorithms. We assume two cases. The first, we know the algorithm solves the noise-free discrete problem and we know its runtime. Second, we do not know the runtime of the algorithm over the noise-free discrete problem.
The work presented here deviates slightly from the results on the previous chapters, but only on the nature of the problems. We can see than applying the same type of principle, reevaluation of the points, can also imply convergent algorithms on discrete noisy problems.
We work here with *runtime* instead of convergence rates for Simple Regret. In discrete optimization there is a preference to use the runtime of the algorithms since it gives easy and straightforward consequences on the performence. Ergo we use here the same concept.
The results presented are based on the journal publication [Akimoto *et al.*, 2015]. This is a joint work with Youhei Akimoto (Shinshu University) and Olivier Teytaud (TAO, INRIA Saclay, now Google).

8.1 Algorithms applied to Discrete Optimization Problems

Discrete optimization usually represent a completely different area, separated from continuous optimization. The discrete nature usually means there is a finite number of possible solutions. This fact can actually be an advantage in small cases when we can check all of the solutions. But discreteness also brings along dimensionality issues: the search space grows rapidly with the dimension. So much that the possible combinations for the solutions explode and it is no longer possible to check all of them.
In this thesis we do not explore the discrete optimization problems specifically. Nonetheless, we offer a result on a model of noisy discrete problems, in order to estimate the runtime of algorithms that can solve the noise-free discrete problem. The work is inspired by the use of Evolutionary

Algorithms (see Chapter 5) over discrete problems.

The development of the studies of Evolutionary Algorithms is based on the study of classical discrete (or combinatorial) problems. These problems have been analyzed from other perspectives therefore we can find in the literature many results that describe the problems. Then the performance of the Evolutionary Algorithms has a base to be compared with: it is possible to place them as more or less efficient than other algorithms.

For instance in [Auger *et al.*, 2011], Chapter 3 offers an analysis of well known discrete optimization problems. The work presented there shows how the analysis of these problems gives us insight on the functioning of Evolutionary Algorithms in general. Even more, the study of these problems shows the use of typical techniques on the analysis of Evolutionary Algorithms.

Another example, is the work on [Rudolph, 1994] who studies the runtime using Genetic Algorithms. Genetic Algorithms constitute a branch of Evolutionary Algorithms and they are characterized by the use of "genes". This means, the points on the search space represent a combination of genes and the aim of the algorithm is either maximize or minimize the fitness of a given gene. The genes are an equivalent of human genes: they can express a finite number of values. Therefore, the Genetic Algorithms are most suited to work with discrete problems. In the case of the work of [Rudolph, 1994], they study the maximization problem of the type

$$\max_{x \in B} F(x), \tag{8.1}$$

assuming $0 \leq F(x) \leq \infty$ for all $x \in B = \{0,1\}^d$ and $F(X) \neq$ constant. Using Markov Chain theory [Rudolph, 1994] proves that canonical simple Genetic Algorithm using crossover and proportional selection do not converge, whilst the elitist version of it does converge.

One difference between the analysis of continuous and discrete problems is that the convergence, on the latter, depends mainly on the dimension of the search space. While as for the continuous problems we are concerned by the amount of function evaluations or iterations.

These thesis will present a modification on the algorithms ready to solve discrete problems in order for them to be also prepared to solve the noisy version of the discrete problems. We will use the same additive noise model as in the continuous problems.

8.2 Context

In this work we provide an upper bound for the runtime of an algorithm adapted to handle noisy functions. The adaptation is done from an algorithm known to be able to solve the deterministic counterpart of the function. We deliver results depending on the previous knowledge on the runtime over the deterministic problem. If we know the runtime of the algorithm that solves the deterministic function, then we adapt it using that information and we derive the runtime of the adapted algorithm. If we do not know the runtime, we adapt the algorithm in each iteration. In both cases the result is an algorithm able to solve the noisy counterpart of the deterministic problem, and we provide the runtime in the noisy case.

This work is focused on discrete optimization problems. The main motivation to develop is its immediate application to Evolutionary Algorithms (EAs). EAs have been successfully applied to discrete optimization problems [Laumanns *et al.*, 2002; Droste *et al.*, 2002; Mühlenbein, 1992; Storch, 2006]. The theoretical analysis of EAs over discrete domain is usually carried out through the use of classical simple EAs over simple optimization problems. Let us define two simple discrete functions used widely for theoretical and experimental analysis: OneMax and LeadingOnes.

Definition 8.1. *Let $d \in \mathbb{N}$. For $x^* \in \{0,1\}^d$ let*

$$\begin{aligned} \mathsf{OM}_{x^*} : \{0,1\}^d &\to \mathbb{N} \\ x &\mapsto \mathsf{OM}_{x^*}(x) = |\{i \in [0,n] \ | x^{(i)} = x^{*(i)}\}| \ . \end{aligned}$$

Let

$$\mathsf{OneMax}_d := \{\mathsf{OM}_{x^*} | x^* \in \{0,1\}^d\} \ . \tag{8.2}$$

Definition 8.2. *Let $d \in \mathbb{N}$. For $x^* \in \{0,1\}^d$ let*

$$\begin{aligned} \mathsf{LO}_{x^*} : \{0,1\}^d &\to \mathbb{N} \\ x &\mapsto \mathsf{LO}_{x^*}(x) = \max\{i \in [0,n] | \forall j \in [1,i], x^{(j)} = x^{*(j)}\} \ . \end{aligned}$$

Let

$$\mathsf{LeadingOnes}_d := \{\mathsf{LO}_{x^*} | x^* \in \{0,1\}^d\} \ . \tag{8.3}$$

For instance, the performance of the algorithms RLS and $(1+1)$-EA over the problems of OneMax and LeadingOnes. The results on the literature roughly indicate that the runtime for OneMax_d is $\theta(d \log d)$ and for $\mathsf{LeadingOnes}_d$ is $\theta(d^2)$ [Auger *et al.*, 2011].

EAs are not only used on deterministic setups. They are naturally robust in the presence of actuator noise[1] [Jong, 1992; Droste, 2004]. Nonetheless, other studies point out that EAs need to be modified to handle additive or multiplicative noise [Jebalia *et al.*, 2011]. The modifications include mainly the use of surrogate models [Ong *et al.*, 2003; Caballero and Grossmann, 2008; Booker *et al.*, 1998; Leary *et al.*, 2004] or some reevaluation technique [Arnold and Beyer, 2006] Surrogate models basically build a model that replaces the need to ask to the real function for evaluations, therefore saving up function evaluations. On the other hand, reevaluation aims to decrease the variance by reevaluating the same point several times and averaging its function evaluations to get a better estimate of the real function evaluation. The reevaluation technique has been studied both theoretically and experimentally. The work presented by us on [Akimoto *et al.*, 2015] focuses on the study of the adaptation of reevaluation techniques from continuous to discrete setup, for functions perturbed by additive noise with constant variance.

We define an optimization algorithm by the sequence of search points it generates. Let `OPT` be the algorithm that solves a class $G = \{g \in G | g : \mathcal{D} \to \mathbb{Z}\}$ of discrete deterministic functions. The sequence generated by `OPT` is defined by

$$x_{n+1} = \mathtt{OPT}(x_1, \ldots, x_n, y_1, \ldots, y_n) \ , \tag{8.4}$$

[1] The actuator noise refers to a perturbation in the search space. In this thesis we focus on perturbation in the image space e.g. the additive noise.

where for every iteration i, $x_i \in \mathcal{D}$ is the search points and $y_i = g(x_i)$ is the function evaluation of point x_i.

We define now K-OPT, the algorithm that is adapted to handle the noisy counterpart of G. Since the functions evaluations in this case are perturbed by noise, K-OPT reevaluates the point x_i several times and then averages the results and assigns this mean as the function evaluation of point x_i. The parameter K represents a sequence that defines the number of reevaluations of the search points. In other words, $K = (k_i)_{i\geq 1}$, where k_i is the number of reevaluations of the search point x_i. The Algorithm K-OPT then can be defined in a similar fashion as the Algorithm OPT in (8.4).

$$x_{n+1} = K\text{-OPT}(x_1, \ldots, x_n, \hat{y}_1, \ldots, \hat{y}_n) \ . \tag{8.5}$$

K-OPT depends on search points and on a mean function evaluations of the each search point. The way to obtain the function evaluations is the main difference between OPT and K-OPT.

8.3 Known runtime, first hitting time

8.3.1 Known runtime, Gaussian noise

This section presents the theorem that states the runtime of K-OPT when we know in advance the runtime of OPT over the deterministic family of functions. The technique and the result explored in this section is similar to the one on [Gutjahr, 2012]. Nonetheless, several differences separate both works. We consider mono objective functions (opposite to [Gutjahr, 2012] that uses bi objective), with a wide class of algorithms and a large family of functions. Also, we consider two noise models: Gaussian and heavy-tail. From the work of [Qian *et al.*, 2014] we know that using reevaluation directly on $(1+1)$-EA to optimize OneMax (see definition 8.1) with Gaussian noise is not enough. In fact, the expected first hitting time increases as the number of reevaluation increases. The result presented here can be applied over K-$(1+1)$-EA, not over $(1+1)$-EA directly.
We define the *runtime* as follows:

Definition 8.3 (Runtime $r(\delta)$ of algorithm OPT to solve G)**.** *The runtime $r(\delta) = r(\delta, G)$ is the number of fitness evaluations needed before* OPT *evaluates the optimum of any function $g \in G$, with probability at least $1 - \delta$.*

We define also $G + \sigma\mathcal{N}$, the noisy counterpart of the family G, solved by OPT. The following Theorem states the runtime of $k_{\mathcal{N}}$-OPT to solve $G + \sigma\mathcal{N}$.

Theorem 8.1. *Assume* OPT *solves G with runtime $r(\delta)$. Let*

$$k_{\mathcal{N}} = \max\left(1, 32\sigma^2\left(\log(2) - \log\left(1 - (1-\delta)^{1/r(\delta)}\right)\right)\right) \ , \tag{8.6}$$

then, $k_{\mathcal{N}}$-OPT solves $G + \sigma\mathcal{N}$ with probability at least $(1-\delta)^2$ and runtime $O\left(r(\delta)\max\left(1, \sigma^2\log\left(\frac{r(\delta)}{\delta}\right)\right)\right)$.

Proof. Let y be the fitness value of a point x and $\hat{y}$ its noisy fitness value. That is to say $\hat{y} = \frac{1}{k_\mathcal{N}} \sum_{i=1}^{k_\mathcal{N}} f(x, \omega_i)$. We will prove that probability of $k_\mathcal{N}$-OPT to make a "mistake" is very small. We will prove then that $k_\mathcal{N}$-OPT obtains exactly the same results as OPT, with probability $(1-\delta)$. Let us compute p the probability of $k_\mathcal{N}$-OPT to make a mistake on one particular point x. In order for the mistake to arise, $\hat{y}$ and y apart by at least $1/4$. In that case $k_\mathcal{N}$-OPT assigns to x a fitness value different than the real one.

$$\begin{aligned}
p &= \mathbb{P}\left(|\hat{y} - y| > \frac{1}{4}\right) , \\
&= \mathbb{P}\left(|\mathcal{N}| > \frac{1}{4}\frac{\sqrt{k_\mathcal{N}}}{\sigma}\right) && \text{using } |\hat{y} - y| \sim \mathcal{N}(0, \sigma^2/k_\mathcal{N}) , \\
&\leq 2\,\mathrm{erfc}\left(\frac{1}{4}\frac{\sqrt{k_\mathcal{N}}}{\sqrt{2\sigma^2}}\right) && \mathrm{erfc} \text{ is the complementary error function} , \\
&\leq 2\exp\left(-\frac{1}{4^2}\frac{\sqrt{k_\mathcal{N}}}{2\sigma^2}\right) && \text{using } \mathrm{erfc}(x) \leq \exp(-x^2) .
\end{aligned}$$

Plugging the definition of $k_\mathcal{N}$ (Equation 8.6) on the latter upper bound for p and some simple algebraic operations, we conclude that $p \leq 1 - (1-\delta)^{\frac{1}{r}}$. Or equivalently,

$$1 - (1-p)^r \leq \delta . \tag{8.7}$$

We note that the left side of Equation 8.7 is exactly the probability of $k_\mathcal{N}$-OPT to make a mistake at least once on the assigning of fitness value for a point x based on its noisy fitness value $\hat{y}$. The difference between the algorithms is that for every evaluation that OPT does, $k_\mathcal{N}$-OPT does $k_\mathcal{N}$ evaluations. Therefore, since OPT finds the optimum of the deterministic problem in runtime r with probability $(1-\delta)$, then $k_\mathcal{N}$-OPT finds the optimum of the noisy problem in runtime $rk_\mathcal{N}$ with probability $(1-\delta)^2$. To obtain the runtime of $k_\mathcal{N}$-OPT we assume that $r \geq 1$ and the general inequality $(1-a) \leq b(1-a^{1/b})$ for $0 \leq a \leq 1$ and $b \geq 1$ we obtain directly from the definition of $k_\mathcal{N}$.

$$k_\mathcal{N} = O\left(\sigma^2 \log\left(\frac{r}{\delta}\right)\right) .$$

Therefore, the runtime of $k_\mathcal{N}$-OPT is $rk_\mathcal{N} = O\left(r \max\left(1, \sigma^2 \log\left(\frac{r}{\delta}\right)\right)\right)$. □

8.3.2 Known runtime, heavy tail noise

We prove in this section that the case with heavy tail noise has a larger upper bound than the Gaussian noise. The proof is very similar, the main difference is that we can no longer compute directly the distribution: we only know that the variance is finite.

Define $G + \Psi$, the noisy counterpart of the family G, solved by OPT. Now Ψ is any random variable with bounded variance σ^2.

Theorem 8.2. *Assume* OPT *solves G with runtime $r(\delta)$. Let*

$$k_\Psi = \max\left(1, 16\sigma^2 \frac{1}{1-(1-\delta)^{1/r(\delta)}}\right) . \tag{8.8}$$

Then, k_Ψ-OPT *solves* $G+\Psi$ *with probability at least* $(1-\delta)^2$ *and runtime* $O\left(r(\delta)\max\left(1,\sigma^2\frac{r(\delta)}{\delta}\right)\right)$.

Proof. We use the same notation as the proof of Theorem 8.1. The proof is analogous, with the following differences. We do not have access to the exact distribution of $\hat{y}-y$, so we use Chebyshev's inequality to upper bound the probability of mistake, p, as follows:

$$p = \mathbb{P}(|\hat{y}-y| \geq \frac{1}{4}) \leq \frac{16\sigma^2}{k_\Psi} \tag{8.9}$$

We plug the definition of k_Ψ on inequality 8.9 to obtain the same bound as in 8.7. Therefore, since OPT finds the optimum of the deterministic problem in runtime r with probability $(1-\delta)$, then k_Ψ-OPT finds the optimum of the noisy problem in runtime rk_Ψ with probability $(1-\delta)^2$. The runtime is $rk_\Psi = O\left(r\max\left(1,\sigma^2\frac{r}{\delta}\right)\right)$. □

8.4 Unknown runtime, general performance measure

The previous section shows how to choose how many reevaluations of each point we should take in order to maintain the convergence of the method with a similar probability. We have used implicitly in the definition of runtime the "first hitting time" criterion, since we count evaluations until we reach the optimum and evaluate it for the first time. This is certainly not the only possible criterion. For instance we might use epsilon distance and be satisfied to have an approximation at a distance at least ε from the real optimum. Another possibility is the first time we reach a "stable" optimum: the number of fitness evaluations t^* necessary such that for all $t \geq t^*$, the current approximation of the optimum is the best, with probability $1-\delta$. Anyway, the thing is that we may want to change what we consider to be a "good enough" approximation, depending on the circumstances.

Therefore we define a general criterion that is either reached or not at iteration T. The formal definition is:

$$\begin{aligned} Q : \cup_{t\in\mathbb{N}}\mathcal{D}^t \times \mathbb{Z}^t &\to \{0,1\} \\ ((x_t),(y_t)) &\mapsto Q\left((x_t),(y_t)\right) \ , \end{aligned} \tag{8.10}$$

where $Q\left((x_t),(y_t)\right) = 1$ if the criterion is reached. Note that the criterion Q can be evaluated at any time t.

We will assume that OPT satisfies a criterion Q for any $T \in \mathbb{N}$ with probability at least $(1-\delta)$ for any objective function $g \in G$. In other words,

$$\forall g \in G, \forall T \in \mathbb{N}, \qquad \mathbb{P}\left[Q\left((x_t)_{t\leq T},(y_t)_{t\leq T}\right)=1\right] \geq 1-\delta \ .$$

We will prove that we can define a sequence that allow the algorithm to reach the criterion. The sequences will be named $K_\mathcal{N} = (k^i_\mathcal{N})_i$ for the case of Gaussian noise, and $K_\Psi = (k^i_\Psi)_i$ for the case of heavy tail.

8.4.1 Gaussian noise

The following theorem proves $K_\mathcal{N}$-OPT reaches the criterion Q with probability at least $(1-\delta)$.

Theorem 8.3. *Assume* OPT *solves G with probability $1-\delta$. Let $\beta > 1$ and*

$$k_{\mathcal{N}}^i = 32\sigma^2 \log\left(\frac{2(i+1)\log(i+1)^\beta}{\delta}\left(\sum_{j=2}^{\infty}\frac{1}{j\log(j)^\beta}\right)\right) . \tag{8.11}$$

Then $K_{\mathcal{N}}$-OPT solves $G+\sigma\mathcal{N}$ with probability at least $(1-\delta)^2$. Additionally, the total number of fitness evaluations up to the i-th iteration is $O(i\log i)$.

Proof. First, note that the term $\sum_{j=2}^{\infty}\frac{1}{j\log(j)^\beta}$ is a Bertrand series and it is convergent as $n\to\infty$ (Lemma 2.7).
We define $u_i = \mathbb{P}(|\hat{y}_i - y_i| \geq 1/4)$ and $v_i = \mathbb{P}(\exists l \leq i : |\hat{y}_i - y_i| \geq 1/4)$. Using the same arguments as the ones used in the proof of Theorem 8.1.

$$\begin{aligned} u_i &\leq 2\exp\left(\frac{-\sqrt{k_{\mathcal{N}}^i}}{32\sigma^2}\right) \\ &\leq \frac{\delta}{(i+1)\log(i+1)^\beta\sum_{j=2}^{\infty}\frac{1}{j\log(i)^\beta}} \qquad \text{using } k_{\mathcal{N}}^i \text{ defined in 8.11.} \end{aligned}$$

Then, by definition of v_i

$$v_i \leq \sum_{j=2}^{i} u_j \leq \sum_{j=2}^{\infty} u_j \leq \delta .$$

Therefore, $K_{\mathcal{N}}$-OPT satisfies the criterion Q with probability at least $(1-\delta)^2$. The total number of function evaluations until iteration i is $\sum_{j=1}^{i} k_{\mathcal{N}}^j \in O\left(\sum_{j=1}^{i}\log i\right) = O(i\log i)$. □

8.4.2 Heavy tail noise

The following theorem constitutes the analogous of the Theorem 8.3, but considering a family of functions with heavy tail noise $G+\Psi$.

Theorem 8.4. *Assume* OPT *solves G with probability $1-\delta$. Let $\beta > 1$ and*

$$k_{\Psi}^i = 16\sigma^2\frac{(i+1)\log(i+1)^\beta}{\delta}\sum_{j=2}^{\infty}\frac{1}{j\log(j)^\beta} .$$

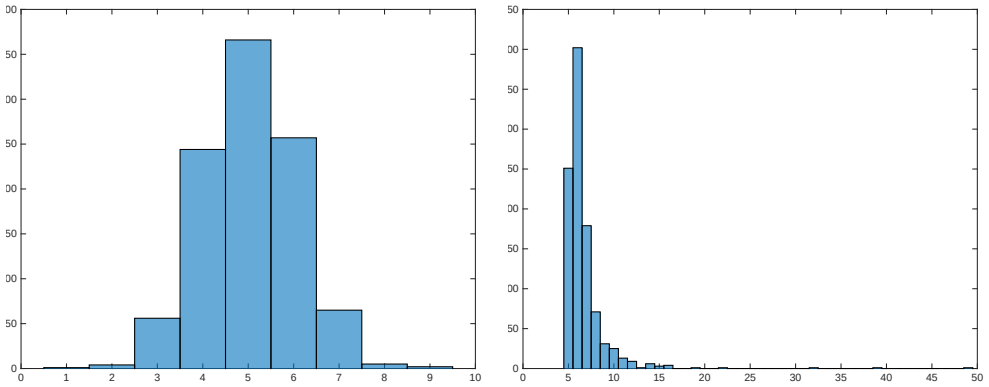
Then K_{Ψ}-OPT solves $G+\Psi$ with probability at least $(1-\delta)^2$. Additionally, the total number of fitness evaluations up to the i-th iteration is $O(i^2(\log i)^\beta)$.

Proof. The proof is analogous to the proof of Theorem 8.3, making the same changes as in the proof of Theorem 8.2. □

8.5 Application

We state here some straightforward consequences on classical algorithms and problems. We apply the theorems presented in this chapter to know their runtime in the noisy case.

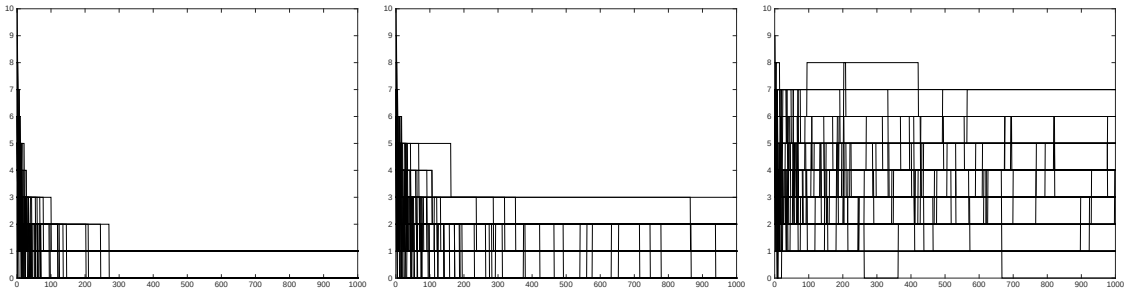
First, we show in Figure 8.1 the effect of Gaussian noise and a heavy tail noise on the function evaluations. The Gaussian noise has parameters $\mu = 0$ and $\sigma = 1$. For the heavy tail noise we use the log-normal distribution with parameters $\mu = 0$ and $\sigma = 1$. We show here the distribution of the value of OneMax with the corresponding noise. We consider 100 noisy function evaluations a specific point in $\mathcal{D} = \{0, 1\}^{10}$.



(a) Gaussian Noise (b) Heavy tail Noise

Figure 8.1: Gaussian and heavy tail, dimension 10, histogram of 100 evaluations over x = (1111100000) for function OneMax. The real function value is OneMax(x) = 5, but we will get perturbed values due to the effect of the noise

In Figure 8.2 we can see the result of the optimization process realized by $(1+1)$-ES with 1/5 rule on the OneMax problem for dimension 10. We run the algorithm 100 times and we plot the difference between the function evaluation of the recommendation and the function evaluation of the optimum. Therefore, we have good recommendations when the value of the y-axis is 0. The x-axis represents the iterations. And note that each of the lines represents a run of the $(1+1)$-ES. The left figure shows how the algorithm finds quickly the optimum (many times under 300 evaluations), even though sometimes it does not reach the optimum. We can compare it with the results in the figures of the center and right side. With the Gaussian noise we can observe how the algorithm struggles to get an optimum and even more, sometimes it finds it and loses it at the next iteration. The results get worse when we consider the heavy tail noise, where the algorithm is sometimes completely wrong. Evidently the noise has a big influence and the algorithm needs some way to cope with the noise.



(a) OneMax deterministic (b) OneMax Gaussian Noise (c) One Max Heavy tail Noise

Figure 8.2: 100 runs of 1+1 EA with 1/5th rule over OneMax dimension 10

We present the results in Table 8.1. The first column refers to the problem and the second column to an algorithm used to solve it. Third and fourth column are the runtime of the algorithm and the reference on the literature. The last two columns represent the runtime on the noisy case: Gaussian and any noise with finite variance. Note that we neglect the precision as a function of δ and the exact computation of K, to focus on the runtime dependency on the dimension d.

Problem	OPT	Runtime	Reference	$k_{\mathcal{N}}$-OPT	k_{Ψ}-OPT
OneMax	$(1+1)$-EA	$O(d \log d)$	[Mühlenbein, 1992]	$O(d(\log d)^2)$	$O(d^2(\log d)^2)$
LeadingOnes	Algorithm in [Droste *et al.*, 2006]	$O(d^2)$	[Droste *et al.*, 2006]	$O(d^2 \log d)$	$O(d^4)$
MaxClique	$(1+1)$-EA	$O(d^5)$	[Scharnow *et al.*, 2004]	$O(d^5 \log d)$	$O(d^{10})$
Sorting	$(1+1)$-EA	$O(d^2 \log d)$	[Storch, 2006]	$O(d^2(\log d)^2)$	$O(d^4(\log d)^2)$

Table 8.1: Runtime comparison of several discrete optimization problems in noise-free and noisy environment. For the runtime in noisy environment we use the results of Theorem 8.1 and 8.2

8.6 Conclusion

We present a modification for algorithms that solve deterministic instances of discrete problems so that they can solve the noisy counterpart of the problem. The noise model considered is additive in two versions: Gaussian and heavy tail. We investigate the runtime of the modified algorithms in comparison to the runtime of the original algorithms. The modified algorithm includes the reevaluation of the search points several times, in order to reduce the variance, and the assignation of the function value by averaging the reevaluations and using a rounding function to assign a discrete function value. If the original algorithm has a known runtime over the deterministic case, we use a number of reevaluation depending on the runtime for the original algorithm, constant in each iteration of the algorithm. For the Gaussian noise we obtain a runtime almost the same as the runtime on the noise free case (extra logarithmic factor) and for the heavy tail noise the runtime is quadratic on the original runtime.

If the original algorithm does not have a known runtime on the deterministic case, we create a sequence of reevaluation number depending on the iteration and a parameter $\beta > 1$. We can also extend the result by considering a general "criterion" that is reached by the algorithm in the

deterministic case. In the case of the modified algorithm, the criterion is also reached in almost the same time as the original case, for Gaussian noise. And the runtime is quadratic for the heavy tail case.

Part III

Conclusion

Chapter 9

Conclusion

We explore in this thesis the study of Noisy Optimization Algorithms. We propose generalizations of popular algorithms and show mathematically several properties such as their convergence order and rate. We support the theoretical results with experiments, using the sphere function with noise for the continuous case. We exhibit a critical analysis on the use accuracy measure in order to compare the performance of algorithms. Finally, we expand the convergence consequences to the case of discrete noisy optimization and determine the convergence rates in that case.
The remainder of this chapter states the conclusions related to each chapter in the Part II: Contributions.

9.1 Chapter 4: Convergence Rates for General Noisy Optimization Algorithm

We propose an algorithm called "Iterative Noisy Optimization Algorithm" (see Algorithm 4.1) that generalizes noisy optimization algorithms by dividing the processes into *search*, *recommendation* and the dependency of some parameters that determine step-size and reevaluation. The algorithm uses the process `SEARCH` to create new search (or sample) points using as input the previous recommendation, the step-size and the evaluation number. The process to generate recommendations is called `OPT` and it uses the previous recommendation and the search points and their function evaluations (possibly reevaluations of the same search point). The algorithm is complemented by a property called "Low Square Error" (LSE), see Definition 4.1, that states basically that recommendations are close to the optimum if search points are.
We prove that two important linesearch algorithms in noisy optimization can be implemented by INOA and they satisfy the LSE: the noisy gradient method and the noisy Newton method. Both the gradient and Newton method presented here use approximations of the gradient and the Hessian, therefore they are zero order methods.
We prove general convergence rates for the noisy Newton method. The analysis yields that with the appropriate parameters, the noisy Newton method can reach convergence rates described in the literature with several algorithms. We also prove a conjecture raised on [Jebalia and Auger, 2008]. We refer the reader to table 4.4 to see a summary of our results and how they compare with the literature.

9.2 Chapter 5: Log-log Convergence for Evolution Strategies

We start by assuming that the result on [Auger, 2005] yields the convergence of ES on the noise-free case for the sphere function. In other words, we take the precise result on [Auger, 2005], which proves the convergence of the sequence $1/n \log(x_n)$ to a constant, and we assume that this constant is negative. The assumption is backed up by experimental work in the literature.
We propose then a scheme of "reevaluation" included in the ES. The reevaluation allows for a good estimation of the real function value, in the case of noisy optimization. The reevaluation depends on a two parameters K and η and the iteration number. We analyze three type of reevaluation: exponential, adaptive and polynomial (see table 5.1 for details). We obtain theoretical order of convergence for exponential and adaptive schemes. We evaluate experimentally the convergence of the polynomial scheme
We obtain in theorem 5.1 that an ES that converges log-linear on the noise-free case for the sphere, converges log-log on the noisy p-sphere (see definition on 5.6) provided that it possesses an exponential reevaluation scheme. Note that we assume to be in presence of scale invariance. We extend the result in theorem 5.2 by getting rid of the scale invariance and now considering and adaptive reevaluation scheme. We also obtain a log-log convergence in this case. For the polynomial scheme we realise several experiments using combinations of the parameters. The experiments suggest log-log convergence.
Note that we cannot say anything about the convergence rate, only that it seems to be $s(SR) \geq -1/2$.

9.3 Chapter 6: Lower Bound on the Convergence Rate of Evolution Strategies

This is a natural continuation of Chapter 5 where we prove a lower bound for the slope of the Simple Regret.
We start by generalizing an optimization algorithm, something similar to the generalization in Chapter 4. But in this case we consider the *search*, *recommendation* and a random element that allows the inclusion of randomized algorithms. We define Evolution Strategies as the algorithms that satisfy special conditions for the search process (see Section 6.3.1) and we denote them by *Simple Evolution Strategies* (Simple ES). The Simple ES considered at this point do not include algorithms that generate search points far away from the optimum. We obtain that the slope of the Simple Regret is lower bounded by $-1/2$ for the sphere function. The important consequence is that for any family that contains the sphere function will be subject to this lower bound when being optimized by a Simple ES.
We provide some experimental verification on figures 6.2 and 6.3 using two ES: UHCMAES and 1+1 ES. We contrast the result of the ES with a noisy linesearch method that uses a one point gradient estimation technique, Shamir Algorithm, figure 6.1. The Shamir Algorithm has a theoretical convergence rate of -1 for the Simple Regret, which can be observed on the experiments presented

in this chapter. On the contrary, the ES exhibit the lower bound of $-1/2$ theoretically exposed before.

9.4 Chapter 7: Performance measure of Noisy Optimization Algorithms

In Chapter 7 we focus on the analysis of the performance measure in noisy optimization algorithms. That is, the object of the study is the performance measure itself and not the performance of an algorithm. The work presented on the previous chapters usually uses the convergence of Simple Regret. We analyze if there is convergence of the sequence and the rate of convergence. With this information we are able to say which algorithm has a better performance over a problem.
But the empirical analysis of algorithms does not allow to obtain the sequence of Simple Regret in noisy optimization. Mainly because the noise prevents us to have access to real function value. We analyze two alternative definitions to Simple Regret and we name them Approximate Simple Regret and Robust Simple Regret. Both of them non-increasing, which can be useful for the use on a real testbed. The important thing should be that the alternative definitions "approximate" well the Simple Regret.
We prove our results using three elements: the literature, conjectures based on the literature and mathematical proofs. The results conclude that the use of these approximations of Simple Regret lead to misleading results. The Robust Simple Regret acts as a lower bound for Simple Regret, but it is not a tight bound. For Evolution Strategies we obtain that the Approximate Simple Regret overestimates their performance, while in the case of noisy linesearch algorithms estimating the gradient we obtain that the Approximate Simple Regret underestimates their performance. We obtain results from an experiment (see figure 7.1) that exhibit the behaviour clearly: the performance algorithms using Simple Regret is not the same as the performance using Approximate Simple Regret.
The results on the poor approximation of Simple Regret by these other measures that attempt to approximate it on real testbeds constitutes an important problem. It is essential to use the adequate performance measures when comparing algorithms. The theoretical analysis and the empirical analysis has to be coherent in order to continue to develop the study of noisy optimization. Unfortunately we are not able to provide such an approximation of Simple Regret, but this work at least shows that the choice of a performance measure on empirical analysis is not obvious and it needs to be justified.

9.5 Chapter 8: Convergence Rates using Reevaluation in Discrete Noisy Optimization

The reevaluation schemes used in Chapter 5 are also applicable to discrete noisy optimization problems in Chapter 8. We assume that a discrete optimization problem (e.g. OneMax, Leadin-

gOnes) is perturbed by some continuous noise (Gaussian or any noise with bounded variance).
As in previous results, we define a general process of optimization `OPT` that takes as input the search points and its function values, and it returns a recommendation point for the current iteration, for the noise-free case. The noisy counterpart of `OPT` will be named K-`OPT`. The parameter $K = (K_i)_{i=1}^{n}$ refers to a sequence of length n that determine the number of reevaluations of the n search points that serve as input.
We analyze the runtime of K-`OPT` in two cases: the runtime of `OPT` over the noise-free problem is known or unknown. In the first case, if we know the runtime for `OPT` in the noise-free discrete problem, say r, then the runtime of K-`OPT` over the noisy counterpart of the discrete problem is $O(r \log r)$. This results holds whether the noise is Gaussian or heavy tail. For the second case, we assume we do not have prior information on the runtime of `OPT` for the noise-free problem. We also assume that `OPT` satisfies a *criterion* (see definition 8.10) with high probability. The criterion extends the runtime concept. We obtain as a result that K-`OPT` also solves (i.e. reaches the criterion) the noisy problem and we give an estimate of the number of function evaluations on iteration i for the case of Gaussian noise. If the noise is any heavy tail random variable, then we have an analogous result, but the number of function evaluations is greater than the Gaussian case.

Bibliography

[Agarwal *et al.*, 2010] A. Agarwal, O. Dekel, and L. Xiao. Optimal Algorithms for Online Convex Optimization with Multi-Point Bandit Feedback. *Conference on Learning Theory-COLT*, pages 28–40, 2010. (Cited on page 26.)

[Akimoto *et al.*, 2015] Y. Akimoto, S. Astete-Morales, and O. Teytaud. Analysis of runtime of optimization algorithms for noisy functions over discrete codomains. *Theoretical Computer Science*, pages 42–50, 2015. (Cited on pages 8, 77 et 79.)

[Arnold and Beyer, 2001] D. V. Arnold and H.-G. Beyer. Investigation of the (mu, lambda)-ES in the presence of noise. In *IEEE Congress on Evolutionary Computation-CEC*, pages 332–339, 2001. (Cited on page 23.)

[Arnold and Beyer, 2002] D. V. Arnold and H.-G. Beyer. Local performance of the (1+ 1)-ES in a noisy environment. *IEEE Transactions on Evolutionary Computation*, pages 30–41, 2002. (Cited on page 58.)

[Arnold and Beyer, 2006] D. Arnold and H.-G. Beyer. A general noise model and its effects on evolution strategy performance. *IEEE Transactions on Evolutionary Computation*, pages 380–391, 2006. (Cited on pages 31, 69 et 79.)

[Astete-Morales *et al.*, 2014] S. Astete-Morales, J. Liu, and O. Teytaud. Log-log Convergence for Noisy Optimization. In *Artificial Evolution*, Lecture Notes in Computer Science, pages 16–28. Springer International Publishing, 2014. (Cited on pages 7, 8 et 47.)

[Astete-Morales *et al.*, 2015] S. Astete-Morales, M.-L. Cauwet, and O. Teytaud. Evolution Strategies with Additive Noise: A Convergence Rate Lower Bound. In *Foundations of Genetic Algorithms-FOGA*, pages 76–84, 2015. (Cited on pages 7, 8 et 57.)

[Astete-Morales *et al.*, 2016a] S. Astete-Morales, M.-L. Cauwet, J. Liu, and O. Teytaud. Simple and cumulative regret for continuous noisy optimization. *Theoretical Computer Science*, pages 12–27, 2016. (Cited on pages 6, 8 et 35.)

[Astete-Morales *et al.*, 2016b] S. Astete-Morales, M.-L. Cauwet, and O. Teytaud. Analysis of Different Types of Regret in Continuous Optimization. *Genetic and Evolutionary Computation-GECCO*, Accepted, 2016. (Cited on pages 8 et 67.)

[Auger and Hansen, 2011] A. Auger and N. Hansen. Theory of evolution strategies: a new perspective. In *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. 2011. (Cited on pages 12 et 27.)

[Auger *et al.*, 2011] A. Auger, A. Auger, and B. Doerr. *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. World Scientific Publishing Co., Inc., 2011. (Cited on pages 53, 54, 78 et 79.)

[Auger, 2005] A. Auger. Convergence results for the (1, lambda)-SA-ES using the theory of phi-irreducible Markov chains. *Theoretical Computer Science*, pages 35–69, 2005. (Cited on pages 7, 47, 51, 60 et 90.)

[Beyer and Schwefel, 2002] H.-G. Beyer and H.-P. Schwefel. Evolution strategies–A comprehensive introduction. *Natural computing*, pages 3–52, 2002. (Cited on page 60.)

[Beyer, 1998] H.-G. Beyer. Mutate large, but inherit small! On the analysis of rescaled mutations in (tilde 1,tilde lambda)-ES with noisy fitness data. *Parallel Problem Solving from Nature–PPSN*, pages 109–118, 1998. (Cited on page 69.)

[Beyer, 2001] H.-G. Beyer. *The Theory of Evolution Strategies*. Natural Computing Series. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001. (Cited on pages 31 et 60.)

[Blum, 1954] J. R. Blum. Multidimensional Stochastic Approximation Methods. *The Annals of Mathematical Statistics*, pages 737–744, 1954. (Cited on page 24.)

[Booker *et al.*, 1998] A. J. Booker, J. E. D. Jr, P. D. Frank, D. B. Serafini, and V. Torczon. Optimization Using Surrogate Objectives on a Helicopter Test Example. In *Computational Methods for Optimal Design and Control*, pages 49–58. 1998. (Cited on page 79.)

[Brooks, 1958] S. H. Brooks. A Discussion of Random Methods for Seeking Maxima. *Operations Research*, pages 244–251, 1958. (Cited on page 27.)

[Caballero and Grossmann, 2008] J. A. Caballero and I. E. Grossmann. An algorithm for the use of surrogate models in modular flowsheet optimization. *AIChE Journal*, pages 2633–2650, 2008. (Cited on pages 23 et 79.)

[Cartis and Scheinberg, 2015] C. Cartis and K. Scheinberg. Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. *arXiv preprint arXiv:1505.06070*, 2015. (Cited on page 20.)

[Chen *et al.*, 2015] R. Chen, M. Menickelly, and K. Scheinberg. Stochastic optimization using a trust-region method and random models. *arXiv preprint arXiv:1504.04231*, 2015. (Cited on page 20.)

[Conn *et al.*, 2000] A. Conn, N. Gould, and P. Toint. *Trust Region Methods*. MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, 2000. (Cited on page 20.)

[Coulom *et al.*, 2011] R. Coulom, P. Rolet, N. Sokolovska, and O. Teytaud. Handling expensive optimization with large noise. In *Foundations of Genetic Algorithms-FOGA*, pages 61–68. ACM, 2011. (Cited on page 23.)

[Coulom, 2012] R. Coulom. CLOP: Confident Local Optimization for Noisy Black-Box Parameter Tuning. In *Advances in Computer Games*, Lecture Notes in Computer Science, pages 146–157. 2012. (Cited on pages 23 et 58.)

[Dang and Lehre, 2015] D.-C. Dang and P. K. Lehre. Efficient Optimisation of Noisy Fitness Functions with Population-based Evolutionary Algorithms. pages 62–68. ACM Press, 2015. (Cited on page 31.)

[Decock and Teytaud, 2013] J. Decock and O. Teytaud. Noisy optimization complexity under locality assumption. In *Foundations of Genetic Algorithms-FOGA*, page 183, 2013. (Cited on pages 58 et 61.)

[Deheuvels, 1983] P. Deheuvels. Strong bounds for multidimensional spacings. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 64(4):411–424, 1983. (Cited on page 69.)

[Droste *et al.*, 2002] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, pages 51 – 81, 2002. (Cited on pages 4 et 79.)

[Droste *et al.*, 2006] S. Droste, T. Jansen, and I. Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of computing systems*, pages 525–544, 2006. (Cited on page 85.)

[Droste, 2004] S. Droste. Analysis of the (1 + 1) EA for a Noisy OneMax. *Genetic and Evolutionary Computation-GECCO*, pages 1088–1099, 2004. (Cited on page 79.)

[Dupac, 1957] V. Dupac. On the Kiefer-Wolfowitz approximation method. *Časopis pro pěstování matematiky*, 1957. (Cited on pages 6, 35 et 44.)

[Elster and Neumaier, 1997] C. Elster and A. Neumaier. A method of trust region type for minimizing noisy functions. *Computing*, pages 31–46, 1997. (Cited on page 20.)

[Fabian, 1967] V. Fabian. Stochastic Approximation of Minima with Improved Asymptotic Speed. *The Annals of Mathematical Statistics*, pages 191–200, 1967. (Cited on pages 6, 22, 24, 25, 35, 44, 58, 64, 73 et 108.)

[Finck *et al.*, 2011] S. Finck, H.-G. Beyer, and A. Melkozerov. Noisy optimization: a theoretical strategy comparison of ES, EGS, SPSA & IF on the noisy sphere. *Genetic and Evolutionary Computation-GECCO*, pages 813–820, 2011. (Cited on page 31.)

[Friedrich *et al.*, 2015] T. Friedrich, T. Kötzing, M. S. Krejca, and A. M. Sutton. The Benefit of Recombination in Noisy Evolutionary Search. In K. Elbassioni and K. Makino, editors, *Algorithms and Computation*, volume 9472, pages 140–150. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. (Cited on page 31.)

[Gentle *et al.*, 2012] J. E. Gentle, W. K. Härdle, and Y. Mori, editors. *Handbook of Computational Statistics*. Springer, 2012. (Cited on page 27.)

[Gutjahr, 2012] W. J. Gutjahr. Runtime analysis of an evolutionary algorithm for stochastic multi-objective combinatorial optimization. *Evolutionary computation*, pages 395–421, 2012. (Cited on page 80.)

[Hansen and Ostermeier, 2001] N. Hansen and A. Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, pages 159–195, 2001. (Cited on pages 29 et 60.)

[Hansen *et al.*, 2009] N. Hansen, A. Niederberger, L. Guzzella, and P. Koumoutsakos. A Method for Handling Uncertainty in Evolutionary Optimization With an Application to Feedback Control of Combustion. *IEEE Transactions on Evolutionary Computation*, pages 180–197, 2009. (Cited on pages 7, 31, 61, 63, 75 et 76.)

[Hansen *et al.*, 2015] N. Hansen, D. V. Arnold, and A. Auger. Evolution Strategies. In *Springer Handbook of Computational Intelligence*, pages 844–871. Springer, 2015. (Cited on page 60.)

[Jebalia and Auger, 2008] M. Jebalia and A. Auger. On multiplicative noise models for stochastic search. *Parallel Problem Solving from Nature–PPSN*, pages 52–61, 2008. (Cited on pages 6, 35, 44 et 89.)

[Jebalia *et al.*, 2007] M. Jebalia, A. Auger, and P. Liardet. Log-linear convergence and optimal bounds for the (1+ 1)-ES. In *International Conference on Artificial Evolution (Evolution Artificielle)*, pages 207–218. Springer, 2007. (Cited on page 31.)

[Jebalia *et al.*, 2011] M. Jebalia, A. Auger, and N. Hansen. Log-linear convergence and divergence of the scale-invariant (1+1)-ES in noisy environments. *Algorithmica*, pages 425–460, 2011. (Cited on pages 31 et 79.)

[Jones *et al.*, 1998] D. R. Jones, M. Schonlau, and W. J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, pages 455–492, 1998. (Cited on page 23.)

[Jong, 1992] K. A. D. Jong. Are genetic algorithms function optimizers. *Parallel Problem Solving from Nature–PPSN*, pages 3–13, 1992. (Cited on page 79.)

[Jägersküpper, 2008] J. Jägersküpper. Lower bounds for randomized direct search with isotropic sampling. *Operations Research Letters*, pages 327–332, 2008. (Cited on page 31.)

[Laumanns *et al.*, 2002] M. Laumanns, L. Thiele, E. Zitzler, E. Welzl, and K. Deb. Running Time Analysis of Multi-objective Evolutionary Algorithms on a Simple Discrete Optimization Problem. *Parallel Problem Solving from Nature–PPSN*, pages 44–53, 2002. (Cited on page 79.)

[Leary *et al.*, 2004] S. J. Leary, A. Bhaskar, and A. J. Keane. A Derivative Based Surrogate Model for Approximating and Optimizing the Output of an Expensive Computer Simulation. *Journal of Global Optimization*, pages 39–58, 2004. (Cited on pages 23 et 79.)

[Matyas, 1965] I. Matyas. Random Optimization. *Automation and Remote Control*, 1965. (Cited on page 30.)

[Mühlenbein, 1992] H. Mühlenbein. How Genetic Algorithms Really Work: Mutation and Hill-climbing. *Parallel Problem Solving from Nature–PPSN*, pages 15–25, 1992. (Cited on pages 79 et 85.)

[Nekrutkin and Tikhomirov, 1993] V. V. Nekrutkin and A. S. Tikhomirov. Speed of convergence as a function of given accuracy for random search methods. *Acta Applicandae Mathematica*, pages 89–108, 1993. (Cited on page 31.)

[Nesterov, 2004] Y. Nesterov. *Introductory Lectures on Convex Optimization*, volume 87 of *Applied Optimization*. Springer, 2004. (Cited on pages 10, 16 et 18.)

[Ong *et al.*, 2003] Y. S. Ong, K. Y. Lum, P. B. Nair, D. M. Shi, and Z. K. Zhang. Global convergence of unconstrained and bound constrained surrogate-assisted evolutionary search in aerodynamic shape design. *IEEE Congress on Evolutionary Computation-CEC*, pages 1856–1863, 2003. (Cited on pages 23 et 79.)

[Powell, 1970a] M. J. Powell. A hybrid method for nonlinear equations. *Numerical methods for nonlinear algebraic equations*, pages 87–114, 1970. (Cited on page 20.)

[Powell, 1970b] M. J. Powell. A new algorithm for unconstrained optimization. *Nonlinear programming*, pages 31–65, 1970. (Cited on page 20.)

[Prugel-Bennett *et al.*, 2015] A. Prugel-Bennett, J. Rowe, and J. Shapiro. Run-Time Analysis of Population-Based Evolutionary Algorithm in Noisy Environments. pages 69–75. ACM Press, 2015. (Cited on page 31.)

[Qian *et al.*, 2014] C. Qian, Y. Yu, Y. Jin, and Z.-H. Zhou. On the effectiveness of sampling for evolutionary optimization in noisy environments. *Parallel Problem Solving from Nature–PPSN*, pages 302–311, 2014. (Cited on page 80.)

[Rechenberg, 1973] I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, 1973. (Cited on page 28.)

[Robbins and Monro, 1951] H. Robbins and S. Monro. A Stochastic Approximation Method. *The Annals of Mathematical Statistics*, pages 400–407, 1951. (Cited on pages 4 et 23.)

[Rolet and Teytaud, 2010a] P. Rolet and O. Teytaud. Adaptive noisy optimization. In *Applications of Evolutionary Computation*, pages 592–601. Springer, 2010. (Cited on pages 6, 35 et 44.)

[Rolet and Teytaud, 2010b] P. Rolet and O. Teytaud. Bandit-based estimation of distribution algorithms for noisy optimization: Rigorous runtime analysis. In *International Conference on Learning and Intelligent Optimization-LION*, pages 97–110. Springer, 2010. (Cited on page 54.)

[Rudolph, 1994] G. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, pages 96–101, 1994. (Cited on page 78.)

[Scharnow *et al.*, 2004] J. Scharnow, K. Tinnefeld, and I. Wegener. The analysis of evolutionary algorithms on sorting and shortest paths problems. *Journal of Mathematical Modelling and Algorithms*, pages 349–366, 2004. (Cited on page 85.)

[Schumer and Steiglitz, 1968] M. Schumer and K. Steiglitz. Adaptive step size random search. *IEEE Transactions on Automatic Control*, pages 270–276, 1968. (Cited on page 29.)

[Schwefel, 1981] H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., 1981. (Cited on page 28.)

[Shamir, 2013] O. Shamir. On the complexity of bandit and derivative-free stochastic convex optimization. *JMLR Workshop and Conference Proceedings*, pages 3–24, 2013. (Cited on pages 6, 7, 25, 26, 35, 44, 57, 58, 62, 64, 72 et 73.)

[Spall, 2000] J. C. Spall. Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE Transactions on Automatic Control*, pages 1839–1853, 2000. (Cited on page 25.)

[Spall, 2003] J. C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. John Wiley & Sons, Inc., 2003. (Cited on pages 20 et 28.)

[Storch, 2006] T. Storch. How randomized search heuristics find maximum cliques in planar graphs. In *Genetic and Evolutionary Computation-GECCO*, pages 567–574, 2006. (Cited on pages 79 et 85.)

[Teytaud and Gelly, 2006] O. Teytaud and S. Gelly. General lower bounds for evolutionary algorithms. *Parallel Problem Solving from Nature–PPSN*, pages 21–31, 2006. (Cited on page 31.)

[Villemonteix *et al.*, 2009] J. Villemonteix, E. Vazquez, and E. Walter. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, pages 509–534, 2009. (Cited on page 23.)

[Yuan, 2000] Y.-x. Yuan. A review of trust region algorithms for optimization. In *international congress on industrial and applied mathematics-ICIAM*, pages 271–282, 2000. (Cited on page 20.)

Chapter 10

Appendix

10.1 Details proof Noisy Gradient based Algorithm satisfies LSE, Section 4.2.1

Proof of equation 4.3:

$$
\begin{aligned}
\mathbb{E}[\hat{g}^{(j)}] &= \mathbb{E}\left[\frac{\hat{y}_{j+} - \hat{y}_{j-}}{2\sigma}\right] && \text{by definition ,} \\
&= \frac{1}{2\sigma}\mathbb{E}\left[\hat{y}_{j+} - \hat{y}_{j-}\right] \ , \\
&= \frac{1}{2\sigma}\mathbb{E}\left[\frac{1}{|Y_{j+}|}\sum Y_{j+} - \frac{1}{|Y_{j-}|}\sum Y_{j-}\right] && \text{by definition ,} \\
&= \frac{1}{2\sigma}\left[\frac{1}{|Y_{j+}|}|Y_{j+}|\mathbb{E}[f(x+\sigma e_j)] - \frac{1}{|Y_{j-}|}|Y_{j-}|\mathbb{E}[f(x-\sigma e_j)]\right] && \text{by definition of } Y_{j+} \text{ and } Y_{j-} \\
& && \text{all evaluations contained in those} \\
& && \text{sets are done over the same points} \\
& && x+\sigma e_j \text{ and } x-\sigma e_j \ , \\
&= \frac{1}{2\sigma}\left[||x+\sigma e_j||^2 - ||x-\sigma e_j||^2\right] \ , \\
&= \frac{1}{2\sigma}\left[\sum_{k=1}^{d}[(x+\sigma e_j)^{(k)}]^2 - \sum_{k=1}^{d}[(x-\sigma e_j)^{(k)}]^2\right] && \text{by definition of } e_j \ , \\
&= \frac{1}{2\sigma}\left[(x^{(j)}+\sigma)^2 - (x^{(j)}-\sigma)^2\right] \ , \\
&= \frac{1}{2\sigma}\left[\left(x^{(j)}\right)^2 + 2x^{(j)}\sigma + \sigma^2 - \left(x^{(j)}\right)^2 + 2x^{(j)}\sigma - \sigma^2\right] \ , \\
&= 2x^{(j)} \ .
\end{aligned}
$$

Proof of equation 4.3:

$$
\begin{aligned}
\mathrm{Var}[\hat{x}^{(j)}] &= \mathrm{Var}\left[\left(x - \frac{1}{2}\hat{g}\right)^{(j)}\right] , \\
&= \frac{1}{4}\mathrm{Var}[\hat{g}^{(j)}] && x \text{ is fixed} , \\
&= \frac{1}{4}\frac{1}{4\sigma^2}\mathrm{Var}[\hat{y}_{j+} - \hat{y}_{j-}] , \\
&= \frac{1}{16\sigma^2}\mathrm{Var}\left[\frac{1}{|Y_{j+}|}\sum f(x+\sigma e_j) - \frac{1}{|Y_{j-}|}\sum f(x-\sigma e_j)\right] , \\
&= \frac{1}{16\sigma^2}\left[\frac{1}{|Y_{j+}|^2}|Y_{j+}|\mathrm{Var}f(x+\sigma e_j) + \frac{1}{|Y_{j-}|^2}|Y_{j-}|\mathrm{Var}f(x-\sigma e_j)\right] && \text{using properties in Lemma 2.3} , \\
&= O\left(\frac{1}{\sigma^2}\frac{1}{r}\sigma^{2z}\right) && \text{using the hypothesis in 4.2,} \\
& && \text{the order of } |Y_{j+}| \text{ and } |Y_{j-}|.
\end{aligned}
$$

10.2 Proof Hessian based Algorithm satisfies LSE, Section 4.3.1

We focus on Newton's method for optimizing a noisy function f such that:

$$
\begin{aligned}
\mathbb{E}f(x,\omega) &= \sum_{1\le j,k\le d} c_{j,k}x^{(j)}x^{(k)} \\
&+ \sum_{1\le j,k,l\le d} b_{j,k,l}x^{(j)}x^{(k)}x^{(l)} + o(\|x\|^3), \text{ with } c_{j,k} = c_{k,j} && (10.1) \\
Varf(x,\omega) &= O(\|x\|^{2z}) \text{ where } z \in \{0,1,2\}. && (10.2)
\end{aligned}
$$

Eq. 10.1 implies that the optimum is $x^* = 0$, which is not a loss of generality as our algorithms are invariant by translation.

We have to introduce some notations. Let M be a matrix, then $t(M)$ will denote the transpose of M and $eig(M)$ the set of the eigenvalues of M. For a given $c > 0$ and a matrix M, we denote by $\mathcal{E}_M^c$ the event : "$\forall v \in eig(M), v \in [c, +\infty)$" and $\overline{\mathcal{E}_M^c}$ the complementary event. h denotes the Hessian of $\mathbb{E}f$ at 0 (so $h = (2c_{j,k})_{1\le j,k\le d}$). Assume that there is some $c_0 > 0$ such that $\mathcal{E}_h^{2c_0}$ holds.

Definition 10.1 (Noisy-Newton method)**.** *Let $x \in \mathbb{R}^d$ and consider some $C > 0$ and $\sigma > 0$ such that :*

$$\|x\| \le C\sigma. \tag{10.3}$$

Define $x_i :=$ `SEARCH`(x, σ, i) and $\hat{x} :=$ `OPT`$((x_i)_{i\in\{1,\dots,\lambda\}}, (y_i)_{i\in\{1,\dots,\lambda\}})$. Compute $\hat{g} = (\hat{g}^{(j)})_{j\in\{1,\dots,d\}}$, approximate gradient at x, by finite difference:

$$\hat{g}^{(j)} := \frac{\hat{\mathbb{E}}f(x+\sigma e_j,\omega) - \hat{\mathbb{E}}f(x-\sigma e_j,\omega)}{2\sigma} ,$$

where $\hat{\mathbb{E}}f(x)$ denotes the average over $\lfloor\lambda/(2d+4d^2)\rfloor$ evaluations at x.

Compute $\hat{h} = (\hat{h}_{j,k})_{1\leq j,k\leq d}$, *approximate Hessian at* x *by finite differences:*

$$\begin{aligned}
g'_{j,k} &:= \frac{\hat{\mathbb{E}}f(x+\sigma e_j+\sigma e_k,\omega)-\hat{\mathbb{E}}f(x-\sigma e_j+\sigma e_k,\omega)}{2\sigma},\\
g''_{j,k} &:= \frac{\hat{\mathbb{E}}f(x+\sigma e_j-\sigma e_k,\omega)-\hat{\mathbb{E}}f(x-\sigma e_j-\sigma e_k,\omega)}{2\sigma},\\
\hat{h}'_{j,k} &:= \frac{g'_{j,k}-g''_{j,k}}{2\sigma},\\
\hat{h} &:= \frac{\hat{h}'+t(\hat{h}')}{2},\ \textit{so that } \hat{h} \textit{ is symmetric.}\\
\textit{Let}\qquad \Delta x &:= \begin{cases} -(\hat{h})^{-1}\hat{g} \textit{ if for a given } c_0>0, \mathcal{E}^{c_0}_{\hat{h}} \textit{ holds,}\\ 0 \textit{ otherwise.}\end{cases}
\end{aligned} \tag{10.4}$$

Using these notations, we define

- *the output of* SEARCH(x,σ,i)*: i.e. the* $x_i =$ SEARCH(x,σ,i) *for* $i\in\{1,\dots,\lambda\}$, *are equally distributed over*

$$E=\{x+\sigma e_j,;\, j\in\{1,\dots,2d\}\}\cup\{x+\sigma e_j+\sigma e_k;\,(j,k)\in\{1,\dots,2d\}^2\}\ ,$$

  so that each of them is evaluated at least $\lfloor\lambda/(2d+4d^2)\rfloor$ *times;*

- $\hat{x}$, *output of* OPT*:* $\hat{x}:=x+\Delta x$.

We first derive the following equalities:

Lemma 10.1. *The approximate gradient verifies:*

$$\hat{g}^{(j)} = \begin{cases} 2\sum\limits_{1\leq k\leq d} c_{j,k}x^{(k)}+\frac{\mathcal{N}_1}{\sqrt{\lambda}\sigma} & \textit{if } b_{j,k,l}=0\ \forall j,k,l,\\ 2\sum\limits_{1\leq k\leq d} c_{j,k}x^{(k)}+\frac{\mathcal{N}_1}{\sqrt{\lambda}\sigma}+O(\sigma^2) & \textit{otherwise.}\end{cases} \tag{10.5}$$

Where $\mathcal{N}_1$ *is an independent noise, with* $\mathbb{E}(\mathcal{N}_1)=0$ *and* $Var(\mathcal{N}_1)=O(\sigma^{2z})$.
The approximate Hessian verifies:

$$\hat{h}_{j,k} = \begin{cases} 2c_{j,k}+\frac{\mathcal{N}_2}{\sqrt{\lambda}\sigma^2} & \textit{if } b_{j,k,l}=0\ \forall j,k,l,\\ 2c_{j,k}+\frac{\mathcal{N}_2}{\sqrt{\lambda}\sigma^2}+O(\sigma) & \textit{otherwise.}\end{cases} \tag{10.6}$$

where $\mathcal{N}_2$ *is an independent noise, independent of* $\mathcal{N}_1$, *with* $\mathbb{E}(\mathcal{N}_2)=0$ *and* $Var(\mathcal{N}_2)=O(\sigma^{2z})$.

Proof. Immediate consequence of the definitions. We get the $1/\sqrt{\lambda}$ part by the averaging over $\Omega(\lambda/(2d+4d^2))$ reevaluations. □

We now give the proof that this pair (SEARCH,OPT) verifies the low squared error assumption (Property 10.1).We consider the case where at least one $b_{j,k,l}\neq 0$, which is the most general case and from which the quadratic case can be easily deduced.

Property 10.1. *If $\sigma^{4-\eta} = O(1/\lambda)$, h is positive definite satisfying $\mathcal{E}_h^{2c_0}$ for a given $c_0 > 0$, B is sufficiently large and σ small enough, then we get $\mathbb{E}\ \|\hat{x}\|^2 = O(\frac{\sigma^\eta}{\lambda})$.*

Proof.

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

where $A_1 \leq (C\sigma)^2$ using Eqs. 10.3 and 10.4; $A_2 = O(\frac{\sigma^\eta}{\lambda\sigma^2})$ by Lemma 10.3 (below); $A_3 = O(\frac{\sigma^\eta}{\lambda})$ by Lemma 10.4 (below). Therefore $E\|\hat{x}\|^2 = O(\frac{\sigma^\eta}{\lambda})$, which is the expected result. □

We have to show Lemmas 10.3 and 10.4. Lemma 10.2 is a first step for proving Lemma 10.3.

Lemma 10.2. *Assume that σ is sufficiently small then $\forall (j,k) \in \{1,\ldots,d\}^2$, $\mathbb{P}(|\hat{h}_{j,k} - h_{j,k}| \geq c_0/d) = O(\frac{\sigma^\eta}{\lambda\sigma^2})$.*

Proof. For short, we use $\hat{h}$ for $\hat{h}_{j,k}$, h for $h_{j,k}$ and c for $c_{j,k}$. By Eq. 10.6, $\mathbb{E}\ \hat{h} = h + O(\sigma)$, so by applying Chebyshev's inequality to $\hat{h}$, we get:

$$\mathbb{P}(|\hat{h} - h| \geq c_0/d) = \mathbb{P}(|\hat{h} - (\mathbb{E}\hat{h} - O(\sigma))| \geq c_0/d) \leq \frac{Var(\hat{h})}{(c_0/d - O(\sigma))^2} ,$$

where

- $Var(\hat{h}) = Var(2c + O(\sigma) + \frac{\omega_2}{\sqrt{\lambda}\sigma^2}) = O(\frac{\sigma^{2z}}{\lambda\sigma^4})$ by Eq. 10.6;
- $(c_0/d - O(\sigma))^2$ is lower bounded, thanks to σ sufficiently small.

Hence, $\mathbb{P}(|h - \hat{h}| \geq c_0/d) = O(\frac{\sigma^{2z-2}}{\lambda\sigma^2}) = O(\frac{\sigma^\eta}{\lambda\sigma^2})$. □

We now use Lemma 10.2 for proving the following:

Lemma 10.3. *Assume that h satisfies $\mathcal{E}_h^{2c_0}$. Then with probability at least $1 - O(\frac{\sigma^\eta}{\lambda\sigma^2})$, $\mathcal{E}_{\hat{h}}^{c_0}$ holds.*

Proof. By Lemma 10.2, with probability $1 - O(\frac{\sigma^\eta}{\lambda\sigma^2})$, $\forall (j,k) \in \{1,\ldots,d\}^2, |\hat{h}_{j,k} - h_{j,k}| \leq c_0/d$. This implies that $(\hat{h} - h)$ has eigenvalues at most c_0 We have also assumed that h has eigenvalues at least $2c_0$.

$\hat{h}$ is a real symmetric matrix, hence it is Hermitian, therefore its eigenvalues are real; using the properties above, for all $x \in \mathbb{R}^d$ (with $\langle .,. \rangle$ the scalar product):

$$\langle \hat{h}x, x\rangle \quad = \quad \langle (\hat{h} - h)x, x\rangle + \langle hx, x\rangle \geq -c_0||x||^2 + 2c_0||x||^2 ,$$

hence $\hat{h}$ has eigenvalues $\geq c_0$. □

We now have to show Lemma 10.4, for concluding the proof of Property 10.1:

Lemma 10.4. *Under assumptions of Lemma 10.3, if $\sigma^{4-\eta} = O(1/\lambda)$, then $\mathbb{E}(\|\hat{x}\|^2|\mathcal{E}_{\hat{h}}^{c_0}) = O(\frac{\sigma^\eta}{\lambda})$.*

Proof.

$$\begin{aligned}\mathbb{E}(||\hat{x}||^2|\mathcal{E}_{\hat{h}}^{c_0}) &= \mathbb{E}(||(\hat{h})^{-1}(\hat{h}x-\hat{g})||^2|\mathcal{E}_{\hat{h}}^{c_0}) \text{ by definition of } \hat{x},\\ &\leq (1/c_0)^2\mathbb{E}(||\hat{h}x-\hat{g}||^2|\mathcal{E}_{\hat{h}}^{c_0}) \text{ by Lemma 10.3.}\end{aligned}$$

$$\begin{aligned}&\text{Under } \mathcal{E}_{\hat{h}}^{c_0},\\ \mathbb{E}(||\hat{h}x-\hat{g}||^2) &= \mathbb{E}\{\sum_{1\leq j\leq d}(\sum_{1\leq k\leq d}\hat{h}_{j,k}x^{(k)}-g^{(j)})^2\},\\ &= d\mathbb{E}\{(O(\sigma^2)+\frac{\mathcal{N}_1}{\sqrt{\lambda}\sigma}+\frac{\mathcal{N}_2}{\sqrt{\lambda}\sigma})^2\} \text{ using Eqs. 10.5, 10.6 and 10.3,}\\ &= O(\sigma^4)+O(\frac{\sigma^{2z}}{\lambda\sigma^2}) \text{ using } \mathbb{E}(\mathcal{N}_1)=\mathbb{E}(\mathcal{N}_2)=0,\\ &\quad Var(\mathcal{N}_1)=Var(\mathcal{N}_2)=O(\sigma^{2z}) \text{ and independence,}\\ &= O(\frac{\sigma^\eta}{\lambda}) \text{ if } \sigma^{4-\eta}=O(1/\lambda),\end{aligned}$$

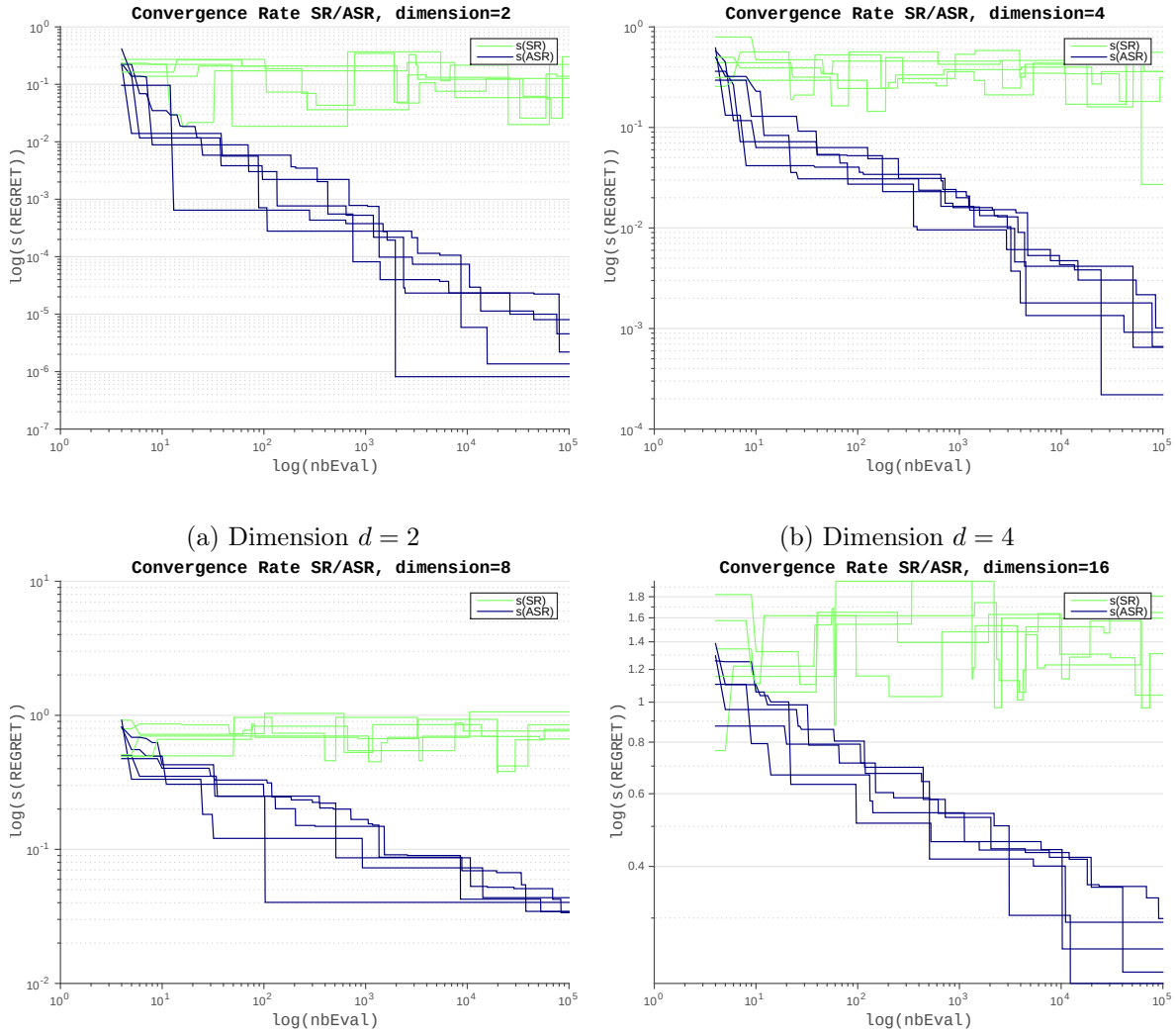
which is the expected result. □

Notice that in Lemma 10.4:

- if $\mathbb{E}f$ is simply quadratic, i.e $\forall(j,k,l)\in\{1,\ldots,d\}^3$, $b_{j,k,l}=0$, the assumption $\sigma^{4-\eta}=O(1/\lambda)$ is unnecessary;
- by taking the expressions of σ and λ given by Alg. 4.1, the condition $\sigma^{4-\eta}=O(1/\lambda)$ is equivalent to $\alpha(4-\eta)\geq\beta$.

10.3 Experiments Random Search, section 7.2

In this section we present experiments for the Random Search algorithm. The function to solve is the noisy sphere. We present in Figure 10.1 the two convergence rates in question: in green the convergence rate of ASR, in blue the convergence rate of SR. We present 5 experiments for each of the dimensions.
In Table 10.1 we present a summary of the experiments. We estimate the convergence rate in average for each of the dimensions, for SR and ASR. In bold we can observe the theoretical bounds presented in Section 7.2 for Random Search: $s(SR)=0$ and $s(ASR)=O(-2/d)$.



(a) Dimension $d = 2$

(b) Dimension $d = 4$

(c) Dimension $d = 8$

(d) Dimension $d = 16$

Figure 10.1: Convergence of the Random Search for SR and ASR. The y-axis represents the logarithm of the corresponding Regret. The x-axis is the logarithm of the number of evaluations.

d	$\mathbf{s(SR)}$	mean $s(SR)$	std $s(SR)$	$\mathbf{s(ASR)}$	mean $s(ASR)$	std $s(ASR)$
2	**0**	-0.0804	0.1086	$\mathbf{-1.000}$	-0.7725	0.2521
4	**0**	-0.1607	0.4126	$\mathbf{-0.500}$	-0.6961	0.3535
8	**0**	-0.0340	0.0623	$\mathbf{-0.250}$	-0.2093	0.1286
16	**0**	-0.0102	0.0725	$\mathbf{-0.125}$	-0.1520	0.0586

Table 10.1: Summary of the experiments in Figure 10.1 for Random Search algorithm. Each row is a specific dimension s. The columns in bold represent theoretical results. The other columns show the mean and standard deviation of the estimated convergence rate. For each curve we compute the slope of the regret and afterwards compute the average and standard deviation over the 5 experiments.

10.4 Proof Theorem 7.1

The following Lemmas will be used on the proof of Theorem (7.1)

Lemma 10.5 (Logarithmic bounds on the quantile of the standard Gaussian variable)**.** *Let $Q(q)$ be q quantile of the standard centered Gaussian, i.e. $\forall q \in (0,1), P(\mathcal{N} \leq Q(q)) = q$. Then $\forall \kappa \geq 1$, $\forall q \in (0,1)$,*

$$1 - \sqrt{-\frac{2}{\kappa}\log(c(\kappa)q)} \leq Q(q) \leq 1 - \sqrt{-2\log(2q)},$$

where $c(\kappa)$ is a constant depending only on κ. In the following, we will denote by $X(q)$ (resp. $Y(q)$) the lower (resp.upper) bound on Q: $X(q) = 1 - \sqrt{-\frac{2}{\kappa}\log(c(\kappa)q)}$ and $Y(q) = 1 - \sqrt{-2\log(2q)}$.

Proof. See http://arxiv.org/pdf/1202.6483v2.pdf. □

Definition 10.2. *We recall that we consider n i.i.d search points $x_1, \ldots, x_n$. Let o and Ω be the standard Landau notations. Let $\varepsilon : \mathbb{N} \setminus \{0\} \mapsto \mathbb{R}$ and $C : \mathbb{N} \setminus \{0\} \mapsto \mathbb{R}$ be two functions satisfying:*

- $\forall n \in \mathbb{N} \setminus \{0\}$, $\varepsilon(n) > 0$ *and* $C(n) > 0$,
- $\forall n \in \mathbb{N} \setminus \{0\}$, $\varepsilon(n) = o(C(n))$,
- $\forall n \in \mathbb{N} \setminus \{0\}$, $\varepsilon(n) = \Omega(1/n)$ *and* $C(n) = \Omega(1/n)$,
- $\forall n \in \mathbb{N} \setminus \{0\}$, $\varepsilon(n) = o(1)$ *and* $C(n) = o(1)$.

Definition 10.3. *With the previous definition of ε and C, consider the set G defined by*

$$G := \{i \in \{1, \ldots, n\}; \|x_i - x^*\| \leq \sqrt{\varepsilon(n)}\}.$$

G is the set of "good" search points, with simple regret better than $\varepsilon(n)$. We denote by $N_G(n)$ the cardinality of G.

Definition 10.4. *Similarly, consider the set B defined by*

$$B := \{i \in \{1, \ldots, n\}; \sqrt{\varepsilon(n)} < \|x_i - x^*\| \leq \sqrt{C(n)}\}.$$

B is the set of "bad" search points, with simple regret bigger than $\varepsilon(n)$, but still not that bad, since the simple regret does not exceed $C(n)$. We denote by $N_B(n)$ the cardinality of B.

Lemma 10.6 (Linear numbers of good and bad points)**.** *There exist a constant $K_d > 0$ such that, with probability at least $1/2$,*

$$N_G(n) < 2K_d n \sqrt{\varepsilon(n)}^d \,, \tag{10.7}$$

and

$$N_B(n) \geq 2K_d n \sqrt{C(n)}^d \,. \tag{10.8}$$

Proof. Proof of Eq. 10.7. Consider a search point x_i. The search points are drawn uniformly at random following the uniform distribution in $[0,1]^d$, then the probability p that $x_i \in G$ is $p = \frac{Vol(B_d(x^*, \sqrt{\varepsilon(n)}))}{Vol([0,1]^d)} = K_d \sqrt{\varepsilon(n)}^d$, where Vol stands for 'volume' and K_d is a constant depending on d only. Therefore the number $N_G(n)$ of good points is the sum of n Bernoulli random variables with parameter $K_d \sqrt{\varepsilon(n)}^d$. The expectation is then $K_d n \sqrt{\varepsilon(n)}^d$. By Markov inequality,

$$\mathbb{P}(N_G(n) \geq 2K_d n \sqrt{\varepsilon(n)}^d) \leq \frac{K_d n \sqrt{\varepsilon(n)}^d}{2K_d n \sqrt{\varepsilon(n)}^d} = \frac{1}{2}.$$

Similarly, $N_B(n)$ is a binomial random variable of parameters n and $p = K_d(\sqrt{C(n)}^d - \sqrt{\varepsilon(n)}^d)$. Then by Chebyshev's inequality, $\mathbb{P}(|N_B(n) - K_d n(\sqrt{C(n)}^d - \sqrt{\varepsilon(n)}^d)| \leq \alpha) \geq 1/2$ by taking $\alpha = \sqrt{2K_d n(\sqrt{C(n)}^d - \sqrt{\varepsilon(n)}^d)(1 - \sqrt{C(n)}^d + \sqrt{\varepsilon(n)}^d)}$. Hence with probability at least $1/2$, $N_B(n) \geq K_d n(\sqrt{C(n)}^d - \sqrt{\varepsilon(n)}^d) + \alpha \geq 2K_d n \sqrt{C(n)}^d$ since $\varepsilon(n) = o(C(n))$. □

We recall that $\forall i \in \{1, \ldots, n\}$, y_i is the fitness value of search point x_i: $y_i = \|x_i - x^*\|^2 + \mathcal{N}_i$, where $\mathcal{N}_i$ is the realisation of a standard centered Gaussian variable.

The following property gives a lower bound on the fitness values of the 'good' points, and an upper bound on the fitness values of the 'bad' points.

Proposition 10.1. *With X and Y as defined in Lemma 10.5, and C as defined in Definition 10.2, there exists some $c \in (0,1)$ such that, with probability at least c,*

$$\inf_{i \in G} y_i \geq X(1/N_G(n)), \tag{10.9}$$

and

$$\inf_{i \in B} y_i \leq C(n) + Y(1/N_B(n)). \tag{10.10}$$

Proof. Consider some Gaussian random variables independently identically distributed $\mathcal{N}_1, \ldots, \mathcal{N}_N$. $\forall\ i \in \{1, \ldots, N\}$, using notation of Lemma 10.5, $\mathbb{P}(\mathcal{N}_i \leq Q(1/N)) = 1/N$, then $\mathbb{P}(\inf_{1 \leq i \leq N} \mathcal{N}_i \leq Q(1/N)) = 1 - \mathbb{P}(\inf_{1 \leq i \leq N} \mathcal{N}_i \geq Q(1/N) = 1 - (1 - 1/N)^N$. The study of the function $x \mapsto 1 - (1 - 1/x)^x$ then shows that $\mathbb{P}(\inf_{1 \leq i \leq N} \mathcal{N}_i \leq Q(1/N)) \in [1 - \exp(-1), 3/4]$ (as soon as $N \geq 2$) Proof of Eq. 10.9.

$$\begin{aligned}
\mathbb{P}(\inf_{i \in G} y_i \geq X(1/N_G(n))) &= \mathbb{P}(\inf_{i \in G} \|x_i - x^*\|^2 + \mathcal{N}_i \geq X(1/N_G(n))) \\
&\geq \mathbb{P}(\inf_{i \in G} \|x_i - x^*\|^2 + \mathcal{N}_i \geq \varepsilon(n) + X(1/N_G(n))) \\
&\geq \mathbb{P}(\inf_{i \in G} \mathcal{N}_i \geq X(1/N_G(n))) \\
&\geq 1 - \mathbb{P}(\inf_{i \in G} \mathcal{N}_i \leq X(1/N_G(n))) \\
&\geq 1 - \mathbb{P}(\inf_{i \in G} \mathcal{N}_i \leq Q(1/N_G(n))) \\
&\geq 1/4.
\end{aligned}$$

Proof of Eq. 10.10.

$$\begin{aligned}\mathbb{P}(\inf_{i\in B} y_i \leq C(n) + Y(1/N_B(n))) &= \mathbb{P}(\inf_{i\in G} \|x_i - x^*\|^2 + \mathcal{N}_i \leq C(n) + Y(1/N_B(n))) \\ &= \mathbb{P}(\inf_{i\in B} \mathcal{N}_i \leq Y(1/N_B(n))) \\ &\geq \mathbb{P}(\inf_{i\in B} \mathcal{N}_i \leq Q(1/N_B(n))) \\ &\geq 1 - \exp(-1)\ .\end{aligned}$$

Hence, with probability at least 1/4, Eqs. 10.9 and 10.10 hold. □

Proof of Theorem 7.1

Proof. Let us assume that the expected simple regret has a slope $-\beta_0 < -\beta$, for some $0 < \beta < 1$: $\varepsilon(SR_n) = O(n^{-\beta_0})$.

We define $\varepsilon(n) = n^{-\beta}$. For some $0 < k < 1$, $\alpha = \beta/k$ and $C(n) = n^{-\alpha}$. ε and C satisfy Definition 10.2.

Lemma 10.6 and Proposition 10.1 implies that there is a $0 < c < 1$ such that with probability c, for n sufficiently large:

- There are much more good points than bad points, i.e.

$$N_G(n) = o(N_B(n)) \tag{10.11}$$

  thanks to Eq. 10.7 and 10.8;

- all good points have noisy fitness at least $X(1/N_G(n)) = 1 - \sqrt{-\frac{2}{\kappa}\log(c(\kappa)/N_G(n))}$;

- at least one bad point has fitness at most $C(n) + Y(1/N_B(n)) = n^{-\alpha} + 1 - \sqrt{-2\log(2/N_B(n))}$;

- therefore (by the two points above, using Eq. 10.11 and the fact that n is big so that $n^{-\alpha}$ is negligeable), at least one bad point has a better noisy fitness than all the good points, and therefore is selected in Lines 7-9 of Alg. 1.

This implies that

$$\begin{aligned}\varepsilon(SR_n) &= \varepsilon(\|x_n - x^*\|^2) = \varepsilon(\|x_n - x^*\|^2 | x_n \in B)\mathbb{P}(x_n \in B) \\ &\quad + \varepsilon(\|x_n - x^*\|^2 | x_n \in G)\mathbb{P}(x_n \in G) \\ &\geq \varepsilon(\|x_n - x^*\|^2 | x_n \in B)\mathbb{P}(x_n \in B) \\ &\geq \varepsilon(n) \times c \\ &\geq cn^{-\beta}.\end{aligned}$$

We have a contradiction, hence $\beta_0 > 0$. □

10.5 Proof of Theorem 7.6

Proof. The upper bounded density is used in Theorem 7.4 for ensuring that no recommendation is never exactly equal to the optimum.

Using the notations in [Fabian, 1967],

$$ASR_n = \min_{n,i,j} F(x^{(i,j)+}(n)) - F(x^*)$$

(or $\min_{n,i,j} F(x^{(i,j)-}(n)) - F(x^*)$, which does not affect the proof), where $x^{(i,j)+}(n)$ is the n^{th} search point. Noting that since there is $s \times d$ evaluations per iterations, $\tilde{x}$ is updated only every $s \times d$ evaluations, so we have $\tilde{x}_n = \tilde{x}_{n+1} = \cdots = \tilde{x}_{n+s\times d-1}$, therefore $x^{(i,j)+}(n) = \tilde{x}_{\lfloor n/s\times d \rfloor} + c_n u_j e_i$. By the convexity of F, the fact that the gradient of F in x^* is 0,

$$F(x^{(i,j)+}(n)) - F(x^*) \geq \frac{\lambda}{2} \|(\tilde{x}_{\lfloor n/s\times d \rfloor} - x^*) + (c_n u_j e_i)\|^2 \ .$$

Similarly, by using the μ-smoothness of F,

$$F(x^{(i,j)+}) - F(x^*) \leq \frac{\mu}{2} \|(\tilde{x}_{\lfloor n/s\times d \rfloor} - x^*) + (c_n u_j e_i)\|^2 \ .$$

Then

$$F(x^{(i,j)+}(n)) - F(x^*) = \Theta(\|(\tilde{x}_{\lfloor n/s\times d \rfloor} - x^*) + (c_n u_j e_i)\|^2) \ .$$

If $\beta > \gamma$, then the main term is the last one and we get a rate -2γ. If $\beta \leq \gamma$, then the main term is the first one and we get a rate -2β.

□