

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Linear Inequality Constraints in Blackbox Optimization

MATHIEU GERVAIS-DUBÉ

Département de mathématiques et de génie industriel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

Mathématiques appliquées

Décembre 2022

POLYTECHNIQUE MONTRÉAL

affiliée à l'Université de Montréal

Ce mémoire intitulé :

Linear Inequality Constraints in Blackbox Optimization

présenté par **Mathieu GERVAIS-DUBÉ**

en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*

a été dûment accepté par le jury d'examen constitué de :

Alain HERTZ, président

Sébastien LE DIGABEL, membre et directeur de recherche

Youssef DIOUANE, membre

DEDICATION

To anyone that will use this work.

ACKNOWLEDGEMENTS

Thanks to Sébastien Le Digabel, who directed me all along my project. He provided quick and valuable answers to all of my questions and requests while trusting me in my ability to complete my project.

I am incredibly grateful to Charles Audet, who helped me throughout this project by guiding me with his extensive knowledge and theoretical expertise.

Thanks to Christophe Tribes who helped me to implement my strategies inside the **NOMAD** software.

Thanks to my parents, who enabled me to learn in a great environment and pursue my graduate studies.

Thanks to my friends in research for inspiring me and with whom it was always a pleasure to have a coffee or have dinner.

Thanks to the GERAD staff for answering my questions and with whom it is always a pleasure to chat.

RÉSUMÉ

L'optimisation de boîtes noires est une branche de l'optimisation permettant de traiter des problèmes où aucune information n'est disponible sur la fonction objectif à minimiser ni sur les contraintes à respecter. Le Mesh Adaptive Direct Search (MADS) est un cadre algorithmique permettant de traiter des problèmes d'optimisation de boîtes noires. Celui-ci possède de fortes propriétés de convergence comparativement à des heuristiques populaires. À ce jour aucune stratégie dédiée n'existe dans MADS ni dans **NOMAD**, son implémentation C++, pour le traitement de contraintes d'inégalité linéaires connues.

Ce mémoire s'intéresse au traitement efficace de ce type de contraintes pour des problèmes d'optimisation de boîtes noires, plus précisément, au cas où ces contraintes sont non relaxables. Quatre stratégies sont proposées: la barrière extrême à priori (APEB), la projection de norme euclidienne (L2), le préconditionnement de bornes (VBP) et l'accrochage aux contraintes (SC). Les trois premières modifient MADS de manière à projeter ou rejeter des points d'évaluation qui sont hors de l'ensemble défini par le système d'inégalités linéaires. Une version préliminaire de ces stratégies est intégrée au logiciel **NOMAD**, une implémentation C++ de MADS.

Les domaines d'intérêts de l'optimisation de boîtes noires sont nombreux. Pour certaines applications, l'évaluation de la fonction objectif et des contraintes représente un coût temporel ou monétaire non négligeable. L'approche proposée permettra éventuellement à **NOMAD** d'offrir de meilleures solutions pour un nombre réduit d'appel à la boîte noire en exploitant les cas où des contraintes d'inégalité linéaires connues sont présentes.

D'une part, l'efficacité des différentes approches proposées est évaluée à l'aide d'expériences sur une simulation de centrale solaire et, d'autre part, de problèmes analytiques de la littérature. Les résultats montrent que la stratégie L2 à base de projection de norme euclidienne performe mieux globalement sur les problèmes considérés tout en conservant les propriétés de convergence de MADS.

ABSTRACT

Blackbox optimization is an optimization branch that deals with problems where information about the objective function and constraints is unavailable. Each time the blackbox is called with a given input, it provides an output that consists of the values of the objective function and the constraints, if they exist. In practice, blackbox calls can have a computationally or a monetary expensive cost. This cost is generally monetary or temporal. Partial information can be available for a given problem, such as the explicit form of some constraints or structural properties. While blackbox methods must be employed, the inherent structure of the problem should be exploited to reduce the cost due to the blackbox calls.

This work proposes strategies to handle efficiently explicit unrelaxable linear inequality constraints within the Mesh Adaptive Direct Search (MADS) method. MADS is an algorithmic framework designed to solve blackbox optimization problems. A dedicated strategy compatible with this framework is developed to handle the aforementioned type of constraints efficiently. This work proposes four strategies: the A Priori Extreme Barrier (APEB), the Snap-To-Constraints (SC), the ℓ_2 Norm Projection (L2), and the variable bounds preconditioning (VBP), to handle them. Their compatibility with MADS ensures they inherit its strong convergence properties.

The strategies are developed in **NOMAD**, a C++MADS implementation. Numerical experiments are run on a set of problems related to both a solar power plant simulation and analytical problems from the literature. Results demonstrate that L2, a strategy that uses a euclidian norm projection on the set formed by the linear inequalities, performs best.

The proof of concept presented in this work implies that future versions of the **NOMAD** software could be able to handle linear inequality constraints to provide better solutions to its users in fewer blackbox evaluations.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
RÉSUMÉ	v
ABSTRACT	vi
TABLE OF CONTENTS	vii
LIST OF TABLES	ix
LIST OF FIGURES	x
LIST OF SYMBOLS AND ACRONYMS	xii
LIST OF APPENDICES	xiii
CHAPTER 1 INTRODUCTION	1
1.1 Blackbox optimization and derivative-free optimization	1
1.2 Characteristics of blackbox optimization problems	3
1.3 Additional notions	4
CHAPTER 2 LITERATURE REVIEW	5
2.1 Direct-search methods	5
2.2 A direct-search method for constrained optimization	9
2.3 Methods for bounds and linearly constrained blackbox optimization	13
CHAPTER 3 PROPOSED STRATEGIES AND MESH PROJECTIONS	15
3.1 The MADS framework for a priori linear inequality constraints	15
3.2 The Segment Search mesh projection (SS)	18
3.3 The proposed strategies	21
3.3.1 A Priori Extreme Barrier (APEB)	21
3.3.2 Snap-To-Constraints (SC)	22
3.3.3 ℓ_2 Norm Projection (L2)	23
3.3.4 Variable bounds preconditioning (VBP)	25

CHAPTER 4	EXPERIMENTAL DESIGN	26
4.1	The problem sets	26
4.2	Data Profiles	28
4.3	Experiments	30
CHAPTER 5	EXPERIMENTAL RESULTS	32
5.1	T-Searches and VBP on the Hock-Schittkowski problems	32
5.2	T-Searches and VBP on the SOLAR problems	38
5.3	The Quadratic Search with T-Searches and VBP on the Hock-Schittkowski problems	40
5.4	Recommandation	42
CHAPTER 6	SOFTWARE DESIGN CHOICES	43
6.1	Providing linear inequality constraints to NOMAD	43
6.2	Solving the strategies' subproblems	44
CHAPTER 7	CONCLUSION AND RECOMMENDATIONS	45
7.1	Summary of works	45
7.2	Limitations	45
7.3	Future research	46
REFERENCES		47
APPENDICES		52

LIST OF TABLES

Table 4.1	The provided set of starting points with their associated function value for each selected problem of the Hock-Schittkowski problem collection.	27
Table 4.2	The provided starting points of the SOLAR problems where the fidelity is set to 0.05 and where the i in f_i denotes the seed index. . .	27
Table A.1	The second set of starting points for the HS problems with their associated value.	53
Table A.2	The third set of starting points for the HS problems with their associated value.	54
Table A.3	The fourth set of starting points for the HS problems with their associated value.	55
Table A.4	The fifth set of starting points for the HS problems with their associated value.	56
Table A.5	Seeds selected for SOLAR	56
Table A.6	Seeds selected for NOMAD	57

LIST OF FIGURES

Figure 2.1	The gridlike mesh and frame for two consecutive iterations of the MADS algorithm.	12
Figure 3.1	Two proposed classes of search for a priori constraints.	17
Figure 3.2	The proposed T-Search strategies.	18
Figure 3.3	The Segment Search is able to project a point $P(t) \in \mathcal{X}$ in $M^k \cap \mathcal{X}$	20
Figure 3.4	The Segment Search fails to project a point $P(t) \in \mathcal{X}$ in $M^k \cap \mathcal{X}$	20
Figure 3.5	The A Priori Extreme Barrier strategy.	21
Figure 3.6	A trial point in black projected onto $\mathcal{X}$ resulting in $P_{\ell_2}(t)$ in red.	24
Figure 4.1	A parameter file for NOMAD where the search strategies are deactivated, and the number of polling directions is set to $2n$	31
Figure 5.1	Data profile of the strategies without Segment Search with tolerance 1E-3 using 1E3 evaluations on the Hock-Schittkowski problems.	32
Figure 5.2	Data profile of the strategies without Segment Search with tolerance 1E-7 using 1E3 evaluations on the Hock-Schittkowski problems.	33
Figure 5.3	Data profile of the strategies with tolerance 1E-7 using 1E3 evaluations on the Hock-Schittkowski problems.	34
Figure 5.4	Data profile of the strategies with tolerance 1E-3 using 1E3 evaluations on the Hock-Schittkowski problems.	35
Figure 5.5	Data profile of the strategies with and without Segment Search with tolerance 1E-7 using 1E3 evaluations on the Hock-Schittkowski problems.	36
Figure 5.6	Data profile of the strategies with and without Segment Search with tolerance 1E-3 using 1E3 evaluations on the Hock-Schittkowski problems.	37
Figure 5.7	Data profile of the strategies with tolerance 1E-3 using 1E3 evaluations on the SOLAR problems.	38
Figure 5.8	Data profile of the strategies with tolerance 1E-7 using 1E3 evaluations on the SOLAR problems.	39
Figure 5.9	Data profile of the strategies mixed with the Quadratic Search with tolerance 1E-3 using 1E3 evaluations on the Hock-Schittkowski problems.	40

Figure 5.10	Data profile of the strategies mixed with the Quadratic Search with tolerance 1E-7 using 1E3 evaluations on the Hock-Schittkowski problems.	41
Figure 6.1	A parameter file for NOMAD where the explicit linear inequality constraints are specified and where the chosen strategy is Snap-To-Constraints.	44

LIST OF SYMBOLS AND ACRONYMS

DFO	Derivative-free optimization
BBO	Blackbox optimization
NOMAD	Nonlinear Optimization by Mesh Adaptive Direct Search
MADS	Mesh Adaptive Direct Search framework
GPS	Generalized Pattern Search algorithm
GSS	Generating Set Search algorithm
CS	Coordinate Search algorithm
LHS	Latin Hypercube Sampling
EB	Extreme Barrier
PB	Progressive Barrier
SS	Segment Search
f_{Ω}	The Extreme Barrier function
$\mathcal{X}$	The explicit set in which the optimization variables reside
x^k	The incumbent solution at iteration k
t	An arbitrary trial point
$Proj(t)$	A trial point projected in $M^k \cap \mathcal{X}$
$P_{\text{name}}(t)$	A trial point projected in $\mathcal{X}$ where name is the name of the used strategy
$L(s) = x^k + (t - x^k)s$	A segment from x^k to t parametrized by parameter $s \in [0, 1]$
$\bar{L}(s)$	A segment from t to x^k parametrized by parameter $s \in [0, 1]$

LIST OF APPENDICES

Appendix A	Generating Points Inside $\mathcal{X}$	52
------------	--	----

CHAPTER 1 INTRODUCTION

In this chapter, blackbox optimization (BBO) and derivative-free optimization (DFO) are defined along with some of their characteristics. The thesis problem is stated, and additional notions are introduced.

1.1 Blackbox optimization and derivative-free optimization

Blackbox optimization and derivative-free optimization (BBO and DFO) [1] deal with optimizing the following model in the unconstrained case

$$\min_{x \in \mathcal{X}} f(x) \tag{1.1}$$

and with Model (1.2) in the constrained case

$$\begin{aligned} \min_{x \in \Omega} f(x) \\ \Omega = \{x \in \mathcal{X} \mid c(x) \leq 0\}, \end{aligned} \tag{1.2}$$

where $\mathcal{X}$ is the set in which the variables relies. Some branches of blackbox optimization can deal with discrete functions, variables, and constraints. However, this work focuses on the case where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, and $c : \mathbb{R}^n \rightarrow \mathbb{R}^p$.

In DFO optimization, information on the derivatives might be available for the objective function and the constraints, but they are not used. In blackbox optimization, there is no information available about the function $f(x)$ and the constraints $c(x)$. While algorithms can be used both for DFO and blackbox optimization, this document will assume that the problems are of the blackbox type. Section 1.2 details the characteristics of such problems.

Examples of blackbox optimization problems that arise in the real world are simulations for engineering designs, such as the design of aircraft engine blades [2] and turbomachine blades [3]. They can also be based on physical experiments, such as the optimal positioning of sensors for snow covers [4]. Many other examples can be found in a recent review [5].

There are cases where the form of the constraints can be explicit, e.g., [6], or where some

information about the structure of the problem is known, i.e., the monotonicity of some of the constraints [7]. When the constraints are explicit, it is possible to include them in $\mathcal{X}$ where $\mathcal{X}$ gives the a priori information on the space of the variables. This problem takes the following form:

$$\begin{aligned} \min_{x \in \Omega} \quad & f(x) \\ \Omega = \{x \in \mathcal{X} : c(x) \leq 0\} \\ \mathcal{X} = \{x \in \mathbb{R}^n : g(x) \leq 0\}, \end{aligned} \tag{1.3}$$

where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \rightarrow \mathbb{R}$, $c : \mathbb{R}^n \rightarrow \mathbb{R}^p$ and $g : \mathbb{R}^n \rightarrow \mathbb{R}^m$.

This thesis adapts an established method to deal with Problem (1.3) where the constraints $g(x)$ are linear inequalities:

$$\begin{aligned} \min_{x \in \Omega} \quad & f(x) \\ \Omega = \{x \in \mathcal{X} : c(x) \leq 0\} \\ \mathcal{X} = \{x \in \mathbb{R}^n : Ax \leq b\}, \end{aligned} \tag{1.4}$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a blackbox function, $c : \mathbb{R}^n \rightarrow \mathbb{R}^p$ are blackbox constraints, $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$.

Section 2 details some families of methods that deal with Problems (1.1), (1.2), (1.3) and (1.4).

Furthermore, Problem (1.4) is further restricted with the following assumptions:

- f can only be evaluated at $x \in \mathcal{X}$. That is, the linear constraints are unrelaxable.
- f is Lipschitz continuous since it is a required hypothesis for the convergence properties of MADS.
- $\mathcal{X}$ is non-empty.
- A starting point $x \in \mathcal{X}$ is provided via an auxiliary method.
- Problem (1.4) has blackbox problem characteristics (see Section 1.2).

These assumptions will stay valid throughout the entire document. Unrelaxable constraints are defined below.

DEFINITION 1.1: UNRELAXABLE CONSTRAINT AS SEEN IN [1]

A constraint $g(x)$ is said to be unrelaxable if it must be satisfied at an evaluation x . Furthermore, when the constraint is violated, a call to a blackbox function may not return a number at all, and if a number is returned, it would be meaningless. That is, the only information obtained from a blackbox call with point $x \notin \mathcal{X}$ is that $x \notin \mathcal{X}$.

According to the unrelaxable constraint Definition 1.1, it is clear that an evaluation outside of $\mathcal{X}$ is superficial and would only spend an evaluation without getting anything in return since $\mathcal{X}$ is known.

The extreme barrier function defined below is a useful function to treat points outside of Ω .

DEFINITION 1.2: THE EXTREME BARRIER FUNCTION AS DEFINED IN [8]

$$f_{\Omega}(x) = \begin{cases} f(x) & \text{if } x \in \Omega \\ \infty & \text{if } x \notin \Omega \end{cases} \quad (1.5)$$

1.2 Characteristics of blackbox optimization problems

Blackbox optimization has distinct characteristics. In this Section, some of them are explained. The proposed methods detailed in Chapter 3 take them into account.

An essential characteristic of blackbox optimization is that generally it is computationally expensive and the time required to evaluate the blackbox is much more than the time required for the mechanism of the optimization algorithm to operate [1]. This means that using the number of blackbox calls required for a particular algorithm to reach a specific value of f is preferred instead of time, as explained in [1]. Indeed, using the number of evaluations is independent of the time required for a machine to return the outputs. Different runs on different machines will always give the same number of calls. It is a desired property for benchmarking comparison.

1.3 Additional notions

In this Section, important notions that are mentioned later are introduced. A positive basis is widely used in direct search methods (see Section 2) and is defined below.

DEFINITION 1.3: POSITIVE BASIS

A set of vectors $V = \{v^1, v^2, \dots, v^{p-1}, v^p\}$, where $v \in V$ is such that $v \in \mathbb{R}^n$, is a positive basis of $\mathbb{R}^n$ if it spans positively $\mathbb{R}^n$. That is, V is a positive basis of $\mathbb{R}^n$ if

$$\forall x \in \mathbb{R}^n, \exists \lambda \in \mathbb{R}_{\geq 0}^p \mid x = \sum_{i=1}^p \lambda_i v^i,$$

Lipschitz continuity for a scalar function is also introduced below.

DEFINITION 1.4: LIPSCHITZ CONTINUITY

A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is Lipschitz continuous on a domain $X \subseteq \mathbb{R}^n$ if and only if there exists a scalar $M > 0$ such that

$$|f(x) - f(y)| \leq M \|x - y\| \quad \forall x, y \in X, \quad (1.6)$$

where $\|\cdot\|$ is the Euclidian norm and $|\cdot|$ is the absolute value norm.

DEFINITION 1.5: LOCALLY LIPSCHITZ FUNCTIONS

A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is locally Lipschitz at $\bar{x}$ if f is Lipschitz on an open ball centered at $\bar{x}$. That is

$$|f(x) - f(y)| \leq M \|x - y\| \quad \forall x, y \in B_r(\bar{x}) := \{z : \|z - \bar{x}\| < r\}, \quad (1.7)$$

where $\|\cdot\|$ is the Euclidian norm and $|\cdot|$ is the absolute value norm.

CHAPTER 2 LITERATURE REVIEW

This chapter reviews different available families of methods for blackbox optimization and specifies a direct-search framework that is used as a baseline to handle Problem (1.4).

Direct-search methods were much developed from 1960 to 1971 [9] and were originally designed for unconstrained optimization. The general idea behind this family of methods is the following. At each iteration, the function is evaluated at multiple points that are around an incumbent solution. If the value of the objective function at one of these points satisfies a decrease condition, then this point replaces the incumbent solution. Section 2.1 provide examples of direct-search methods.

Model-based algorithms are another popular family of algorithms for BBO, and they can be separated into two categories: trust-region methods and descent methods. The former iteratively finds trial points by taking the solution of a minimization subproblem that optimizes a model of the objective function over a ball (the trust region). The algorithmic parameters are updated according to the information gathered by evaluating the function [10]. Basically, the latter category estimates, at each iteration, a descent direction using a modeled gradient and does a line search along this direction [10].

Heuristics such as the Nelder-Mead [11] algorithm and different evolutionary algorithms (Particle Swarm Algorithm [12], Simulated annealing [13], etc.) have been found to have some practical use and be competitive in terms of performance on some problems. However, they generally have weak convergence properties because of the absence of convergence-related stopping criteria, as explained in [1]. Because of this, they are oftentimes oriented toward global optimization.

2.1 Direct-search methods

One of the first known direct-search methods for unconstrained optimization is the **Coordinate Search** (CS), also known as compass search and is presented in [14]. The Coordinate Search selects trial points at each iteration k according to the scaled unit coordinate directions $D^k = \pm\delta^k e_i$ with $i \in [1, n]$. The step-length δ^k is reduced when no trial points improve the value of $f(x_k)$. Convergence to stationary points can be proven if f is continuously differentiable and uniformly continuous since at least one direction in D^k must be a descent direction if $\nabla f(x_k) \neq 0$.

Another mentionable direct-search method is the **Generalized Pattern Search** (GPS) [15]. The GPS is a two-step method in which the first step allows a local exploration around the incumbent solution, and the second step (search step) is a broad exploration in the variable space. The set of directions P^k called a pattern is obtained at each iteration by a basis matrix $B \in \mathbb{R}^{n \times n}$ and a generating matrix $C^k \in \mathbb{Z}^{n \times p}$ such that $P^k = BC^k$, $p > 2n$ and both matrices have rank n to span the entire $\mathbb{R}^n$ space. The generating matrix C^k can change from iteration to iteration. The GPS algorithm can also treat bound and linear inequality constraints, as seen in [16]. It adapts its pattern to the geometry of the constraints that are close (ϵ -active) to the iterate x^k . Its convergence to KKT points under some hypothesis is established. The construction of generators (generators of tangent and normal cones of the linear bounds within distance) is detailed in the nondegenerate case. Following this work, [17] presents a way to identify redundant constraints using the projection method in the case where the set of active constraints is degenerate and construct the poll directions in the degenerate case. The set of active constraints B_k can be classified as nondegenerate (where B_k is full rank), degenerate redundant (where B_k is not full rank and its constraints are linearly dependent), and degenerate non-redundant (where B_k is not full rank and its constraint are linearly independent such as the top of a pyramid in a 3D space).

A generalization of the GPS and CS algorithm is the **Generating Set Search** (GSS) introduced in [14]. The first difference between GSS and CS is that it allows a sufficient decrease condition $f(t) \leq f(x_k) + \rho(\delta^k)$, where [14] gives the following condition for the forcing function ρ :

DEFINITION 2.1: CONDITION ON ρ TAKEN FROM [14]

$$\begin{cases} \rho = 0, \text{ or} \\ \rho \text{ is continuous, } \rho(t) = o(t) \text{ as } t \downarrow 0 \text{ and } \rho(t_1) \leq \rho(t_2) \text{ for } t_1 < t_2 \end{cases} \quad (2.1)$$

Where $o(t)$ as $t \downarrow 0$ means that $\lim_{t \rightarrow 0} \frac{\rho(t)}{t} = 0$. If $\rho = 0$, then it is the simple decrease condition.

GSS also allows δ^k to be increased when the sufficient decrease condition is met. Furthermore, GSS generalizes the directions by introducing $\mathcal{G}^k$ and $\mathcal{H}^k$. They, respectively, are the generating set and additional directions. $\mathcal{G}^k$ is required and must be a positive basis of $\mathbb{R}^n$. The ℓ_2 norm of $h \in \mathcal{H}^k$ must be bounded below by $\beta_{\min}$ and the ℓ_2 norm of $g \in \mathcal{G}^k$ must be bounded by $\beta_{\min}$ below and above by $\beta_{\max}$. Together, they form the search directions $\mathcal{D}^k = \mathcal{G}^k \cup \mathcal{H}^k$ at each iteration k . Finally, a lower bound $\kappa_{\min}$ on the generating set cosine measure $\kappa(\mathcal{G}_k)$ is introduced such that $\kappa(\mathcal{G}_k) > \kappa_{\min}$.

The cosine measure is defined below.

DEFINITION 2.2: THE COSINE MEASURE ON A SET OF DIRECTIONS TAKEN FROM [14]

The cosine measure $\kappa(D)$ of a finite set of directions $D = \{d_1, d_2, \dots, d_i, \dots, d_p\} \mid d_i \in \mathbb{R}^n\}$ is

$$\kappa(D) = \min_{v \in \mathbb{R}^n} \max_{d \in D} \frac{v^T d}{\|v\| \|d\|}$$

In some way, this measure indicates, in the worst-case scenario, how far the search direction can be from the direction of maximal descent. The greater $\kappa_{\min}$ is, the smaller the worst-case scenario can be. Intuitively, as κ tends to 1, the order p of a set of directions D grows indefinitely, and this can be cumbersome computationally. Therefore, a user must consider this tradeoff when increasing the value of $\kappa_{\min}$. In conclusion, one can consider GSS as a framework since there is much freedom in the chosen directions $\mathcal{D}^k$ and the shrinking and expanding parameters of the step-length.

The GSS framework can be adapted for linearly constrained optimization using the same idea as in GPS [14]. That is, the generating set will have to conform to the geometry of the linear constraints with some condition on the generating set to ensure that a descent direction is possible.

Algorithm 1 presents a variant of original GSS algorithm that is adapted to the constrained case.

Remark 2.1.1. *Conditions 8.4 and 8.8 referenced in the GSS framework (see Algorithm 1) can be found in [14] for more details, but they basically ensure that a descent direction is possible for the generating set $\mathcal{D}_k$.*

Algorithm 1: The GSS framework for linearly constrained optimization taken from [14].

Initialization

$f : \mathbb{R}^n \rightarrow \mathbb{R}$ // The given function
 $x_0 \in \Omega$ // Initial guess in Ω
 $\Delta_{\text{tol}} > 0$ // The step-length convergence tolerance
 $\Delta_0 > \Delta_{\text{tol}}$ // Initial value of the step-length parameter
 $\phi_{\text{max}} \geq 1$ // An upper bound on the expansion factor
 $\theta_{\text{max}} < 1$ // Upper bound on the contraction parameter
 $\rho(t)$ // The forcing function defined previously (see Definition 2.1)
 $\beta_{\text{max}} \geq \beta_{\text{min}} \geq 0$ // upper and lower bounds on the lengths of the
 vectors in any generating set
 $\kappa_{\text{min}} > 0$ // A lower bound on the cosine measure of any generating set
 $\epsilon^* > 0$ // The tolerance to choose the active constraints

Step 1

Let $\mathcal{D}_k = \mathcal{G}_k \cup \mathcal{H}_k$. Choose the generating set $\mathcal{G}_k$ such that it respects Conditions 8.4 and 8.8 from [14] and such that $\beta_{\text{min}} \leq \|d\| \leq \beta_{\text{max}}$ for all $d \in \mathcal{G}_k$ and $\kappa(\mathcal{D}_k) \geq \kappa_{\text{min}}$, and $\mathcal{H}_k$ is a finite (possibly empty) set of additional search directions such that $\beta_{\text{min}} \leq \|d\| \forall d \in \mathcal{H}_k$.

Step 2

If there exists $d_k \in \mathcal{D}_k$ such that $f_{\Omega}(x_k + \Delta_k d_k) < f(x_k) - \rho(\Delta_k)$, then:
 - Set $x_{k+1} = x_k + \Delta_k d_k$ (change the iterate).
 - Set $\Delta_{k+1} = \phi_k \Delta_k$, where $1 \leq \phi \leq \phi_{\text{max}}$ (optionally expand the step-length control parameter).

Step 3

Otherwise:
 - Set $x_{k+1} = x_k$ (no change to the iterate).
 - Set $\Delta_{k+1} = \theta_k \Delta_k$ where $0 < \theta_k < \theta_{\text{max}} < 1$ (contract the step-length control parameter).
 - If $\Delta_{k+1} < \Delta_{\text{tol}}$, then **Terminate**

2.2 A direct-search method for constrained optimization

The **Mesh Adaptive Direct Search** (MADS) is a class of algorithms and part of the direct-search method family. Audet and Dennis introduce it in [18] to deal with constrained nonsmooth optimization using the extreme barrier. While the GPS algorithm previously stated could guarantee that the generalized directional derivatives were nonnegative for only a finite set of directions, MADS allows all the possible directions to be visited asymptotically so that the generalized directional derivatives are nonnegative for infinitely many directions. Therefore, the limit point found by the GPS algorithm does not necessarily converge to a Clarke stationary point defined below.

DEFINITION 2.3: CLARKE STATIONARY POINT DEFINED IN [19]

A Clarke stationary point is a point such that the Clarke generalized directional derivative is ≥ 0 for all directions in $\mathbb{R}^n$.

$$f^\circ(x, v) = \lim_{y \rightarrow x, t \downarrow 0} \sup \frac{f(y + tv) - f(y)}{t} \geq 0, \forall v \in \mathbb{R}^n,$$

where $f^\circ(x, v)$ is the generalized directional derivative for a given direction v .

Unlike GPS, MADS converges to a Clarke stationary point. Additionally, it guarantees convergence for constrained problems, while GPS does not. That is, it converges to a constrained Clarke stationary point defined as follows:

DEFINITION 2.4: CONSTRAINED CLARKE STATIONARY POINT DEFINED IN [19]

A constrained Clarke stationary point is a point $\hat{x}$ such that the Clarke generalized directional derivative at $\hat{x}$ is ≥ 0 for all directions d in $T_\Omega^H(\hat{x})$, where T_Ω^H is the hyper-tangent cone. That is,

$$f^\circ(\hat{x}, v) = \lim_{y \rightarrow \hat{x}, t \downarrow 0} \sup \frac{f(y + tv) - f(y)}{t} \geq 0, \forall v \in T_\Omega^H(\hat{x}).$$

MADS can guarantee a dense asymptotic set of directions by introducing the frame and mesh notion defined on the next page.

DEFINITION 2.5: MESH AND MESH SIZE PARAMETER DEFINITION TAKEN FROM [1]

Let $G \in \mathbb{R}^{n \times n}$ be invertible and $Z \in \mathbb{Z}^{n \times p}$ be such that the columns of Z form a positive spanning set for $\mathbb{R}^n$. Define $D = GZ$. The mesh of coarseness $\delta^k > 0$ generated by D , centered at the incumbent solution $x^k \in \mathbb{R}^n$, is defined by

$$M^k := \{x^k + \delta^k Dy : y \in \mathbb{N}^p\} \subset \mathbb{R}^n,$$

where δ^k is called the mesh size parameter.

DEFINITION 2.6: FRAME AND FRAME SIZE PARAMETER DEFINITION TAKEN FROM [1]

Let $G \in \mathbb{R}^{n \times n}$ be invertible and $Z \in \mathbb{Z}^{n \times p}$ be such that the columns of Z form a positive spanning set for $\mathbb{R}^n$. Define $D = GZ$. Select a mesh size parameter $\delta^k > 0$ and let Δ^k be such that $\delta^k \leq \Delta^k$. The frame of extent Δ^k generated by D , centred at the incumbent solution $x^k \in \mathbb{R}^n$, is defined by

$$F^k := \{x \in M^k : \|x - x^k\|_\infty \leq \Delta^k b\},$$

with $b = \max\{\|d'\|_\infty : d' \in \mathbb{D}\}$ and where Δ^k is called the frame size parameter.

where $\mathbb{D}$ is the set of directions taken from the columns of D .

Algorithm 2 describes MADS. In short, each iteration k has a search and poll step. The poll step is entered if the search step fails. In the search step, any method can be used to choose a finite set of trial points on the mesh to be evaluated. In the poll step, trial points must be chosen in the frame such that the set of their directions form a positive basis for $\mathbb{R}^n$. At each iteration, the frame size parameter either shrinks or expands according to successes or failures. The parameter update ensures that δ^k goes to zero faster than Δ^k , and it allows the polling directions to be asymptotically dense. In practice, the positive spanning matrix $D = GZ$ defined in Definition 2.5 uses coordinate directions. This generates an orthogonal mesh. Furthermore, the polling directions must be randomly generated to ensure that they are asymptotically dense.

Figure 2.1 shows $n + 1 = 3$ trial points in red around the incumbent solution x^k in black. In the first iteration, all the trial points t are such that $f(t) \geq f(x_k)$. In the next iteration, the frame size parameter is updated $\Delta^1 = \tau \Delta^0 = (0.5)(1) = 0.5$, and the mesh size parameter is updated accordingly $\delta^1 = (0.5)^2 = 0.25$. New trial points are chosen according to directions that positively span $\mathbb{R}^2$.

Algorithm 2: The MADS framework for constrained optimization taken from [1].

0. Initialization

$x_0 \in \Omega$	// A starting point in Ω
$\Delta^0 \in (0, \infty)$	// The initial frame size parameter
$D = GZ$	// A positive spanning matrix
$\tau \in (0, 1)$ with $\tau \in \mathbb{Q}$	// mesh size adjustment parameter
$\epsilon_{\text{stop}} \in [0, \infty)$	// A stopping tolerance
$k = 0$	// An iteration counter

1. Parameter Update

└ Set the mesh size parameter to $\delta^k = \min\{\Delta^k, (\Delta^k)^2\}$.

2. Search

└ **if** $f_\Omega(t) < f_\Omega(x^k)$ *for some* t *in a finite subset* S^k *of* M^k **then**
 └ Set $x^{k+1} \leftarrow t$ and $\Delta^{k+1} \leftarrow \tau^{-1}\Delta^k$ and go to **4**.

3. Poll

└ Select a positive spanning set $\mathbb{D}_\Delta^k$ such that $P^k = \{x^k + \delta^k d : d \in \mathbb{D}_\Delta^k\}$ is a subset of F^k of extent Δ^k .
 if $f_\Omega(t) < f_\Omega(x^k)$ *for some* $t \in P^k$ **then**
 └ Set $x^{k+1} \leftarrow t$ and $\Delta^{k+1} \leftarrow \tau^{-1}\Delta^k$.
 otherwise
 └ Set $x^{k+1} \leftarrow x^k$ and $\Delta^{k+1} \leftarrow \tau\Delta^k$.

4. Termination

└ **if** $\Delta^{k+1} \geq \epsilon_{\text{stop}}$ **then**
 └ Increment k and go to **1**.
 otherwise
 └ Stop

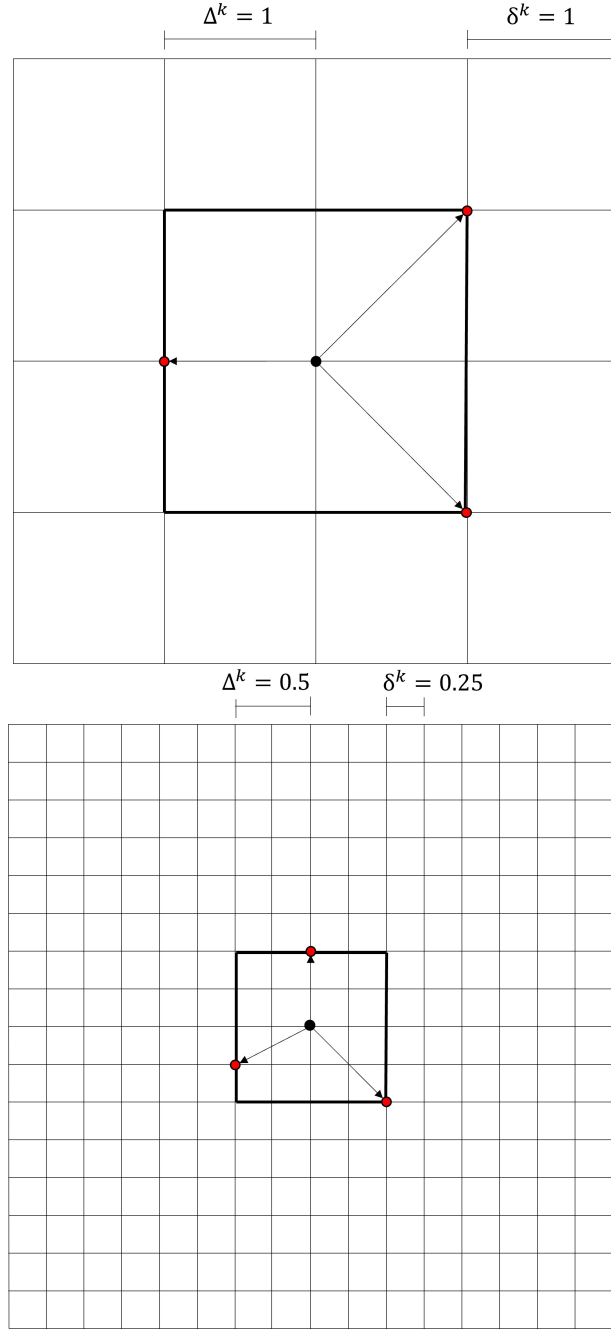


Figure 2.1 The gridlike mesh and frame for two consecutive iterations of the MADS algorithm.

MADS can be considered as a framework since many of its parts can be modified, such as the search step, the parameter update, the chosen directions for the poll step, and the mesh adjustment parameter. Different implementations of the MADS algorithm are developed in the **NOMAD** software. It is currently at its fourth version (see [20]).

2.3 Methods for bounds and linearly constrained blackbox optimization

This Section reviews other proposed methods in the literature for bound-constrained and linearly constrained problems.

Already discussed is the convergence of the MADS algorithm for nonsmooth constrained problems. It can also handle explicit linear equalities and simple bounds on x efficiently if it is adapted consequently. The work by [21] uses the MADS method combined with projection in the null space of A in order to have a set of candidates Z that respects linear inequality constraints. It then excludes the candidates outside of the simple bounds.

This work [12] by Vicente and Vaz handles general linear inequalities in a global optimization perspective using the PSwarm solver. In this solver, a population of particles is moving in the feasible space by using a modified version of coordinate search that defines positive generators according to the tangent cone of ϵ -active constraints in the poll step and dampens the displacement in the search step. The method requires initial feasible points. The initial positions of the particles are generated by creating an ellipsoid of maximum volume in the feasible set constrained by the inequalities using the work of [22]. The extreme barrier function is applied in the poll step to handle infeasible trial points. Vicente and Vaz [12] also present a collection of linearly constrained optimization problems.

Work done by Diouane and al. in [23] proposes the use of an evolution strategy (ES) in the case of general unrelaxable constraints. For the linear inequality case, it proposes two approaches. The first approach uses ES in conjunction with the projection of unfeasible points in the feasible region bounded by general constraints. The second approach arises for the linear inequality cases. It adapts the directions used to generate new offsprings to the local geometry of the constraints. For the case of general linear inequality constraints, an ℓ_1 norm projection into the feasible space is used. For the case where there are only simple bounds, the ℓ_2 projection is used since its computation is trivial. These approaches are compared on two sets of problems: blackboxes with general inequality constraints and blackboxes that have only simple bounds.

In [24], a global optimization algorithm is presented by partitioning the feasible space defined by simple bounds $\ell_i \leq x_i \leq u_i$. Each element of the partition of $\ell_i \leq x_i \leq u_i$ is called a tile, and a random representative is chosen for each tile. At each iteration, tiles are cut, and new representatives are generated. After a number of iterations, the algorithm is restarted with a different configuration according to a stochastic process.

Recently, [25] proposed a probabilistic approach in direct-search methods that selects random polling directions in the tangent cone in combination with a sufficient decrease condition. In

other words, the polling directions must not necessarily be the positive generators of the active tangent cone but rather randomly generated in it. Another condition on the random polling direction is that they should be probabilistic descent. The probabilistic descent condition is defined in [26], and the same article shows that the choice of taking polling directions derived from uniformly generated vectors on the unit sphere has good performance numerically.

In trust region methods, [27] describes LINCOA [28], an algorithm for problems with linear constraints that use quadratic models to approximate the next best iterate and where linear constraints are selected locally by defining an active set of constraints. These active constraints are added to the trust-region subproblem. This algorithm uses interpolation with the least-Frobenius norm condition [29] to update quadratic models that mimic the objective function $Q(x) \approx f(x)$. Likewise, [30] proposes LCOBYQA, a trust region method based on NEWUOA [31] for unconstrained optimization and that differs from it in the construction of its subproblems.

An inner approximation approach is presented in [32] for structured blackbox optimization problems where the solution is expressed in terms of a linear combination of points called atoms and where the feasible set is a convex hull. The goal is to optimize large-scale problems and express the solution using a combination of a small number of atoms. The scheme consists of adding/deleting atoms and finding candidates in reduced problems. The authors prove that this method, called ORD, has global convergence to stationary points under the requirement of smoothness of the objective function.

This current document explores an approach that does not use ϵ -active constraints. Instead, it uses the convergence properties of MADS in conjunction with the extreme barrier function with methods in the search step dedicated to linear inequality constraints. Infeasible polling points are treated a priori by the extreme barrier function and carried out in the search step, where they are projected on the mesh and in the feasible domain by means of a projection technique. Furthermore, the task of selecting initial feasible starting points is generated by Latin Hypercube Sampling (see Appendix A) on the variable bounds or by applying MADS to a cost function that incorporates unrespected constraints. The next chapter details the approaches taken.

CHAPTER 3 PROPOSED STRATEGIES AND MESH PROJECTIONS

In this chapter, the MADS framework is specified for Problem (1.4) by adapting the poll and search steps accordingly. Then, a mesh projection heuristic is defined, ensuring that the trial points transferred from the poll step to the search step are on the mesh. The convergence for these strategies is inherited from the properties of the MADS framework. For the rest of the document, the word "search" refers to a method that generates trial points in the search step of MADS.

3.1 The MADS framework for a priori linear inequality constraints

The original MADS algorithm does not make a difference between linear inequality constraints in $\mathcal{X}$ and the blackbox constraints $c(x)$. However, **NOMAD** can manage simple bounds by projecting the trial points to the bounds [33]. Furthermore, if linear inequality constraints can be reduced to linear equality constraints, then the MADS framework can be adapted [21]. Otherwise, evaluating trial points outside of $\mathcal{X}$ spends useless evaluations of the blackbox.

A minor modification is proposed for MADS. Each trial point is verified with respect to the linear inequality constraints before the evaluation of the objective function. If it is outside of $\mathcal{X}$, it is not evaluated and saved. A heuristic *Proj* attempts to project a subset of the saved points on the mesh and in $\mathcal{X}$. If it succeeds, it sends these projected points to the search step to be evaluated. This idea is presented in Algorithm 3. Modifications to the original framework are in light green. Blackbox evaluations occur at the end of the search step and at the beginning of the poll step. The set P^k is constructed in the pre-poll step instead of the poll step for practicality.

The following remarks give more information about the proposed framework detailed in Algorithm 3.

Remark 3.1.1. S^k is defined as a subset of $M^k \cap \mathcal{X}$ since any trial point in the search step should reside in $\mathcal{X}$ to save useless blackbox evaluations. It is the user of MADS that has to choose points to form the set S^k .

Remark 3.1.2. T^k is allowed to be empty such as the case where all the trial points in $\mathcal{T}^k$ are rejected a priori using the extreme barrier (see Section 3.3.1). The two other proposed strategies use $T^k = \mathcal{T}^k$. See 3.3.2 and 3.3.3.

Algorithm 3: The proposed variation of the MADS framework to manage linear constraints that defines $\mathcal{X}$.

0. Initialization

$x_0 \in \Omega$ // A starting point in Ω
 $\Delta^0 \in (0, \infty)$ // The initial frame size parameter
 $D = GZ$ // A positive spanning matrix
 $\tau \in (0, 1)$ with $\tau \in \mathbb{Q}$ // mesh size adjustment parameter
 $\epsilon_{\text{stop}} \in [0, \infty)$ // A stopping tolerance
 $k = 0$ // An iteration counter
 $\mathcal{T}^0 = \emptyset$ // Set of trial points that are outside of $\mathcal{X}$

1. Parameter Update

Set the mesh size parameter to $\delta^k = \min\{\Delta^k, (\Delta^k)^2\}$.

2. Pre-Poll

Select a positive spanning set $\mathbb{D}_{\Delta}^k$ such that $P^k = \{x^k + \delta^k d : d \in \mathbb{D}_{\Delta}^k\}$ is a subset of F^k of extent Δ^k where F^k and M^k respectively are the frame and the mesh defined in 2.6 and 2.5.

if some $t \in P^k$ is outside of $\mathcal{X}$ **then**
 $P^k \leftarrow P^k \setminus t$ (Remove t from P^k).
 $\mathcal{T}^k \leftarrow \mathcal{T}^k \cup t$ (Add t to $\mathcal{T}^k$).

3. Search

Set S^k a finite subset of $M^k \cap \mathcal{X}$ (see Remark 3.1.1).

Select a possibly empty subset T^k such that $T^k \subseteq \mathcal{T}^k$ (see Remark 3.1.2).

for $t \in T^k$ **do**
 Apply a projection $Proj$ on t where $Proj : \mathbb{R}^n \rightarrow \mathbb{R}^n$ tries to satisfy $Proj(t) \in M^k \cap \mathcal{X}$.
 if $Proj(t) \in M^k \cap \mathcal{X}$ **then**
 $S^k \leftarrow S^k \cup Proj(t)$ (Add $Proj(t)$ to S^k).
if $f_{\Omega}(t) < f(x^k)$ for some t in S^k **then**
 Set $x^{k+1} \leftarrow t$ and $\Delta^{k+1} \leftarrow \tau^{-1} \Delta^k$ and go to **5**.

4. Poll

if $f_{\Omega}(t) < f(x^k)$ for some $t \in P^k$ **then**
 Set $x^{k+1} \leftarrow t$ and $\Delta^{k+1} \leftarrow \tau^{-1} \Delta^k$.
otherwise
 Set $x^{k+1} \leftarrow x^k$ and $\Delta^{k+1} \leftarrow \tau \Delta^k$.

5. Termination

if $\Delta^{k+1} \geq \epsilon_{\text{stop}}$ **then**
 Increment k and go to **1**.
otherwise
 Stop.

Two projections are proposed in 3.3.2 and 3.3.3. If such heuristics can not satisfy $Proj(t) \in M^k \cap \mathcal{X}$ after a finite number of operations, then neither t nor $Proj(t)$ is added to S^k .

The task is then to define projections applied to $t \in \mathcal{T}^k$ that derive trial points to be added to S^k . This kind of strategy is of class **T-Search** since it is applied on the set $\mathcal{T}^k$.

The other option is to construct trial points in S^k without using $\mathcal{T}^k$. This search type is denoted as a **S-Search** strategy. Naturally, both classes of searches can be used. Figure 3.1 and 3.2 present a global view of these options and the proposed strategies. In Figure 3.1, A Priori MADS (APMADS) denotes all possible methods that could be invented inside the MADS framework to treat a priori constraints. VBP is a strategy that is introduced in Section 3.3.4.

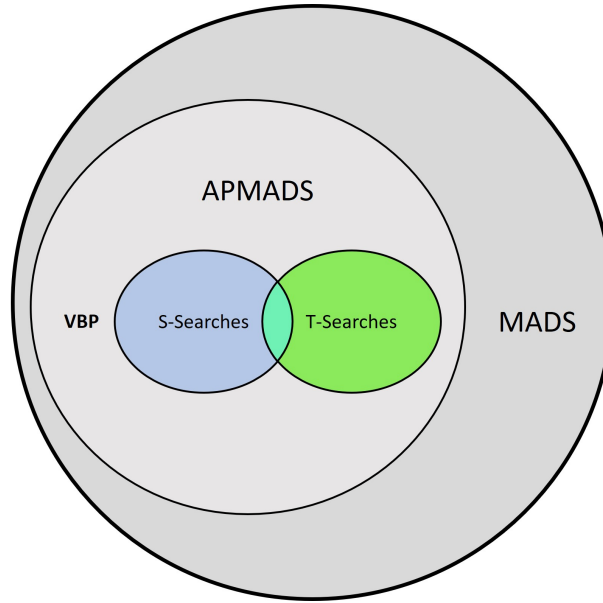


Figure 3.1 Two proposed classes of search for a priori constraints.

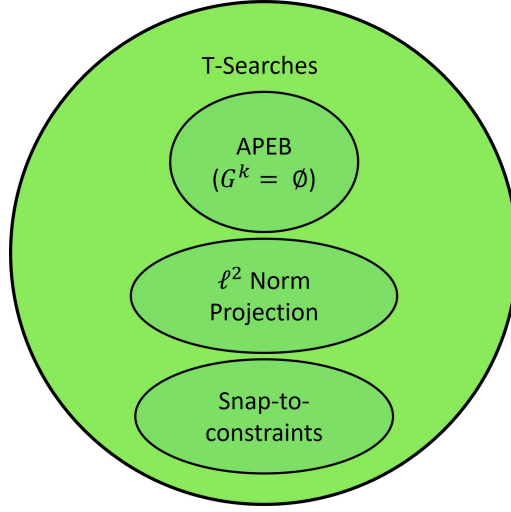


Figure 3.2 The proposed T-Search strategies.

3.2 The Segment Search mesh projection (SS)

Suppose a trial point (t) was projected inside $\mathcal{X}$ by any means resulting in $P(t)$. The next step is to project it on the mesh while ensuring it stays in $\mathcal{X}$. The projection on the mesh is necessary to secure the convergence properties of MADS. Many heuristics could be defined to do this task. In this section, the Segment Search heuristic is introduced. It is used in Section 3.3.3 and 3.3.2 to complete the projections in $\mathcal{X}$.

The Segment Search does a search along the segment starting from $P(t) \in \mathcal{X}$ and goes to x^k . It uses a ratio r for each $P(t)$ defined as follows.

$$r = \left\lceil 2 \max_{i \in \{1, \dots, n\}} \frac{|v_i|}{\delta^k} \right\rceil \quad (3.1)$$

where v_i is an element of the vector $v = P(t) - x^k$ and $|\cdot|$ is the absolute value norm. If the mesh size parameter δ^k is scaled for each variable, then this ratio can be modified by replacing δ^k by δ_i^k . Another option is to take $r = \left\lceil 2 \frac{\|P(t) - x^k\|}{\delta^k} \right\rceil$ where $\|\cdot\|$ is the Euclidian norm. If the variables are scaled, δ^k is replaced by $\delta_{\min}^k$ where $\delta_{\min}^k$ is the smallest δ_i^k associated with element i .

The Segment Search detailed in Algorithm 4 moves along the segment using steps on the parameter of the segment. At each step, it rounds a point along the segment with respect to δ^k and checks if it still belongs in $\mathcal{X}$. The rounding operation to multiples of δ^k guarantees

that the result will be in M^k since the mesh is gridlike. If no point is found, the result is not added to S^k . Figure 3.3 and 3.4 give two-dimension examples of success and failure for the heuristic. For the experiments, the maximum number of tries allowed (MT) is set to 1000.

Algorithm 4: The Segment Search.

Initialization

```

    Given  $P(t) \in \mathcal{X}$                                 // A trial point  $t \in \mathcal{T}^k$  projected in  $\mathcal{X}$ .
    Given  $x^k$                                           // The incumbent solution.
    Given  $\delta^k$                                        // The mesh size parameter.
     $\omega = P(t)$                                        // A point along the segment.
     $v = x^k - P(t)$                                      // Direction and norm of the segment.
     $r$           // A ratio for the number of tries as defined in Equation (3.1)
     $MT$          // The maximum number of tries allowed
     $NoS = \max \left\{ 1, \min\{r, MT\} \right\}$            // Number of steps to try.
     $\alpha = \frac{1}{NoS}$                                // Step length parameter for the segment search.

for  $i = \{0, \dots, NoS - 1\}$  do
     $y = \text{round}(\omega)_{\delta^k}$                          // Round  $\omega$  to nearest multiples of  $\delta^k$ .
    if  $y \in \mathcal{X}$  then
        Stop and Return  $\omega = y$  the point projected on the mesh.
    Otherwise
         $\omega = \omega + \alpha v$ 
Return no solution found.

```

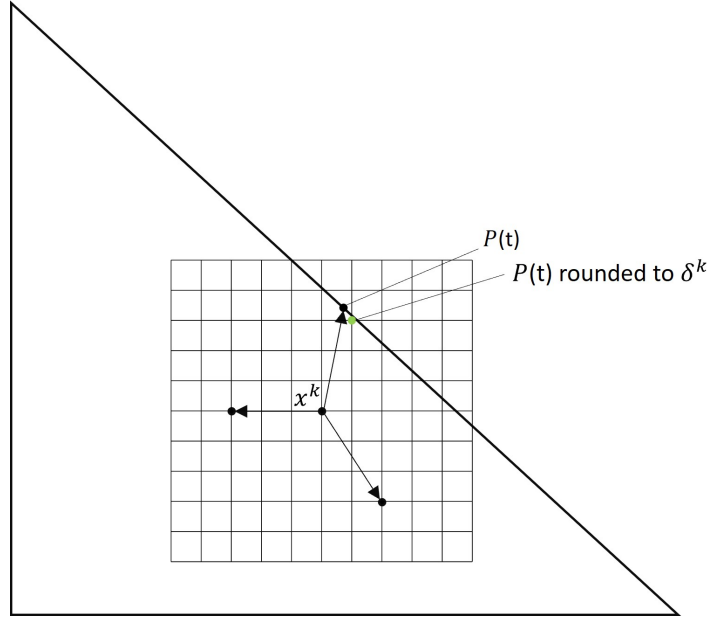


Figure 3.3 The Segment Search is able to project a point $P(t) \in \mathcal{X}$ in $M^k \cap \mathcal{X}$.

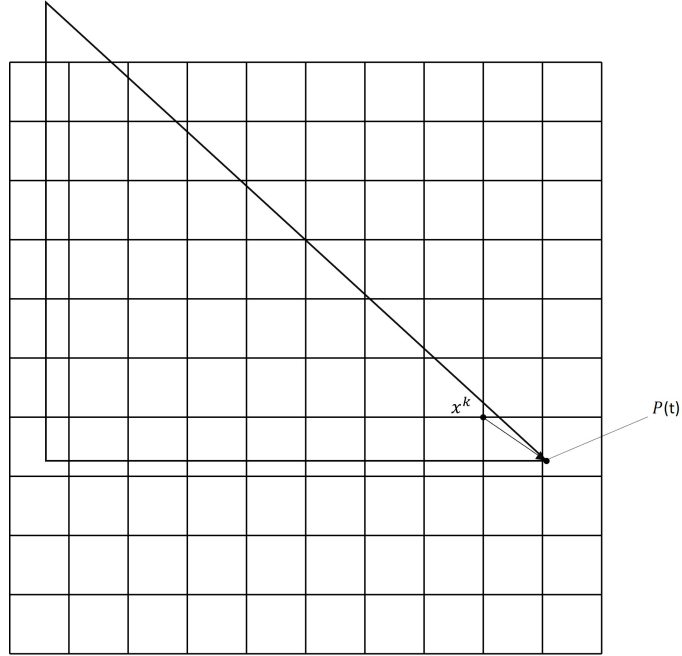


Figure 3.4 The Segment Search fails to project a point $P(t) \in \mathcal{X}$ in $M^k \cap \mathcal{X}$.

3.3 The proposed strategies

In this section, the different proposed strategies are defined and classified using the framework presented in Algorithm 3 for a priori constraints.

3.3.1 A Priori Extreme Barrier (APEB)

Recall that in MADS with the extreme barrier, if unknown constraints are unrelaxable, the infeasible points are set to ∞ . So to speak, they are discarded since they can not replace the incumbent solution. Therefore, it is possible to discard all the trial points $t \notin \mathcal{X}$ without evaluating the function. The A Priori Extreme Barrier (APEB) is this simple strategy. Figure 3.5 presents its idea.

Setting $T^k = \emptyset$ in Algorithm 3 corresponds to this basic strategy. It is the simplest T-Search strategy possible.

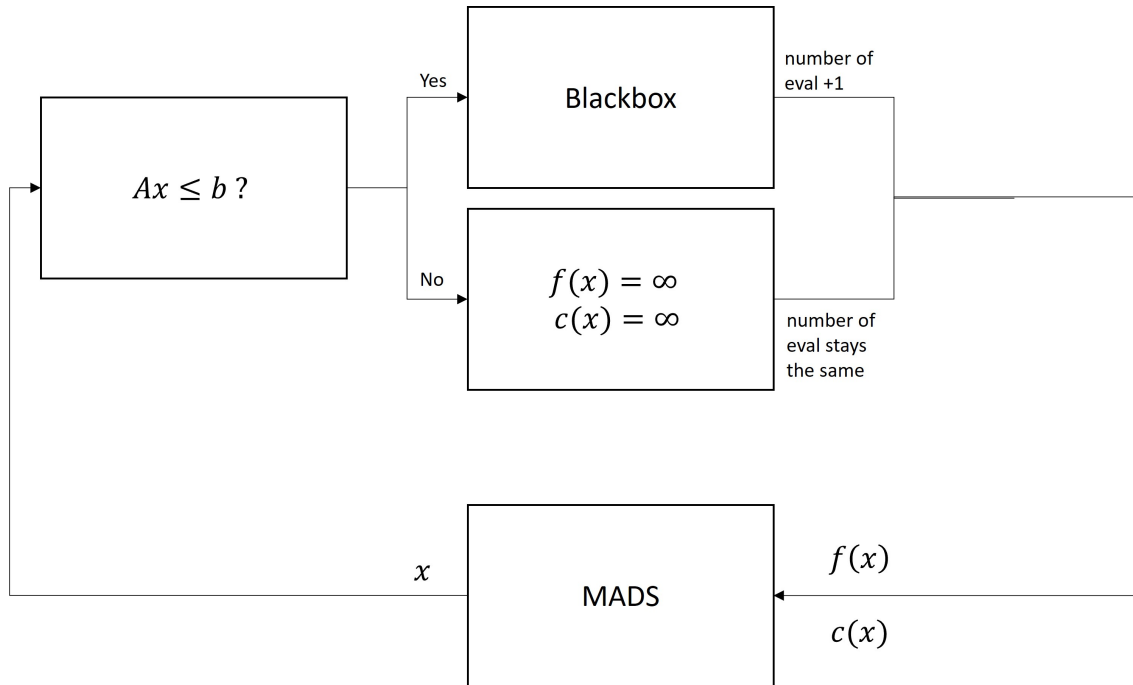


Figure 3.5 The A Priori Extreme Barrier strategy.

3.3.2 Snap-To-Constraints (SC)

The Snap-To-Constraints strategy is a two-phase strategy. At each iteration, it is applied to a trial point $t \in \mathcal{T}^k$. In the first phase, it considers a segment that goes from the frame center x^k to a trial point t . Then, the intersections of the segment with the hyperplanes $A_i x = b_i$ are calculated. The intersection that is the closest to x^k is chosen to replace the trial point t . In its second phase, the Segment Search heuristic defined in Section 3.2 is applied to the solution of the first phase to project it on the mesh.

Suppose the segment is parametrized as $L(s) = x^k + (t - x^k)s$ and $s \in [0, 1]$ is a parameter. The solution of the first phase is obtained by calculating the smallest value of the parameter s for the intersections of the segment with the hyperplanes. Each value s_i associated with hyperplane i is calculated in the following way:

$$s_i = \frac{b_i - A_i(x^k)}{A_i(t - x^k)} \quad \forall i \in I, \quad (3.2)$$

where $I = \{1, 2, \dots, m\}$ is the set of indexes associated with lines in $Ax \leq b$. Then $P_{sc}(t) = x^k + s_{\min}(t - x^k)$. The value of $s_{\min}$ is obtained by calculating all the distances s_i and taking the minimum as detailed in Algorithm 5.

Algorithm 5: Finding the intersection of the segment $L(s)$ with the first encountered hyperplane.

Initialization

Set $s_{\min} = 1$	// Minimum parameter of the intersection
Given x^k	// The position of the incumbent solution
Given t	// A trial point in the poll step

for $(A_i, b_i) \in (A, b)$ **do**

if $s_i = \frac{b_i - A_i(x^k)}{A_i(t - x^k)} < s_{\min}$ and $s_i \geq 0$ then	
$s_{\min} = s_i$	

Set the intersection $P_{sc}(t) = x^k + s_{\min}(t - x^k)$

The formula for the value s_i associated with each hyperplane is derived in the following way:

$$A_i(L(s_i)) = b_i \quad (3.3)$$

$$A_i(x^k + (t - x^k)s_i) = b_i \quad (3.4)$$

$$A_i x^k + A_i(t - x^k)s_i = b_i \quad (3.5)$$

$$s_i = \frac{b_i - A_i(x^k)}{A_i(t - x^k)} \quad (3.6)$$

Computing $P_{sc}(t)$ concludes the first phase. The second phase of this procedure is to project $P_{sc}(t)$ on the mesh using the Segment Search defined in 3.2.

3.3.3 ℓ_2 Norm Projection (L2)

The ℓ_2 Norm Projection strategy is in two phases. In its first phase it takes $t \notin \mathcal{X}$ and project it into $\mathcal{X}$ using the smallest euclidian distance. This operation is denoted as $P_{\ell_2}(t)$. In this strategy, all the trial points outside of $\mathcal{X}$ are projected in $\mathcal{X}$. In its second phase it tries to project on the mesh the result of the first phase using the Segment Search. It is a T-Search with $T^k = \mathcal{T}^k$.

Figure 3.6 shows a two-dimension example of the ℓ_2 projection.

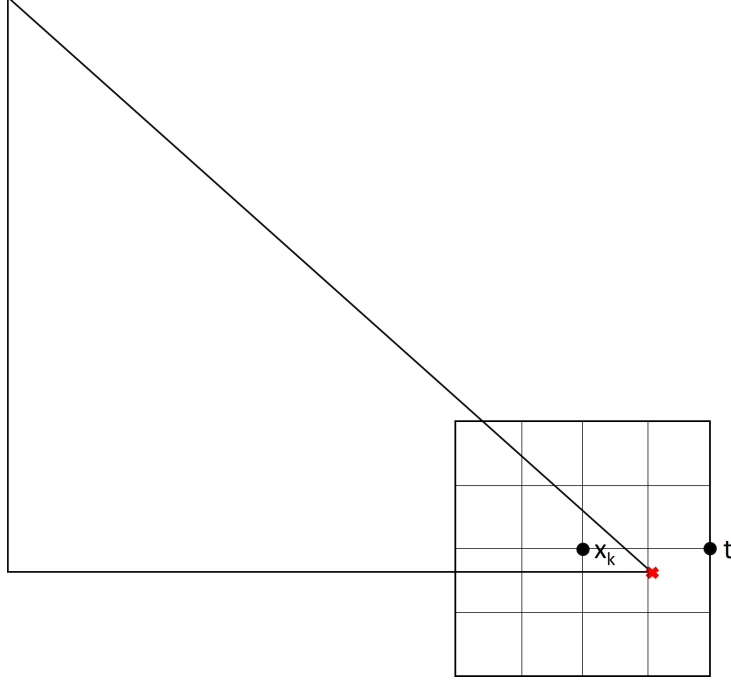


Figure 3.6 A trial point in black projected onto $\mathcal{X}$ resulting in $P_{\ell_2}(t)$ in red.

The difficulty in this approach is computing the projection. The projection of a point $t \in \mathbb{R}^n$ on the convex set formed by the hyperplanes can be modeled by this minimal ℓ_2 norm problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \|t - x\|_2 \\ \text{s.t.} \quad & Ax \leq b \end{aligned} \tag{3.7}$$

This problem is equivalent to solving the following quadratic programming problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & q(x) = \frac{1}{2}x^T Gx + x^T c \\ \text{s.t.} \quad & Ax \leq b \end{aligned} \tag{3.8}$$

with $G = I$ and $c = -t$. The projected point $P_{\ell_2}(t)$ is obtained this way.

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & q(x) = \frac{1}{2}x^T x - x^T t \\ \text{s.t.} \quad & Ax \leq b \end{aligned} \tag{3.9}$$

This can be solved by a quadratic programming solver or by a non-linear programming solver.

3.3.4 Variable bounds preconditioning (VBP)

This strategy consists of finding bounds on each variable and providing them to **NOMAD**, which is already able to use this information [33]. **NOMAD** snaps the point to the simple bounds using the snap-to-bounds procedure, but it does not project them on the mesh. For this reason, the snap-to-bound procedure is not classified as a T-Search. The upper and lower bounds for a given variable x_i can be found by solving the following linear programming problems:

$$\ell_i = \min_{x \in \mathbb{R}^n} x_i \tag{3.10}$$

$$\text{s.t.} \quad Ax \leq b \tag{3.11}$$

$$u_i = \max_{x \in \mathbb{R}^n} x_i \tag{3.12}$$

$$\text{s.t.} \quad Ax \leq b \tag{3.13}$$

A linear programming method can solve these problems before launching MADS.

CHAPTER 4 EXPERIMENTAL DESIGN

Taking into account the mentioned characteristics of blackbox optimization (see Section 1.2), the experimental design to compare the different strategies is detailed in this Section. The fourth version of **NOMAD** (see [20]) is used as the implementation of MADS for the experiments.

4.1 The problem sets

Two problem sets are chosen to compare the strategies. The first consists of problems from the Hock-Schittkowski collection originally designed to test nonlinear programming codes [34]. These problems are analytical functions that are considered as blackboxes in this work. Only problems with linear inequality constraints with or without bounds are used for the tests. The problems are HS21, HS24, HS35, HS36, HS37, HS44, HS76, HS86 and HS105. This collection is useful to test blackbox algorithms since they have distinct characteristics such as badly scaled objectives, multiple local solutions, and nonsmooth model function [34]. For example, HS21 has a badly scaled quadratic objective. HS76, HS86, and HS105 are classified as practical problems since their exact solution is not known [34].

Furthermore, authors in [34] provide a starting point for each problem. It is preferable to add new ones to avoid biases stemming from them. Therefore, four new feasible starting points are generated for each of the chosen problems in the Hock-Schittkowski collection. The ones from the Hock-Schittkowski collection are given in Table 4.1, and the rest is provided in Appendix A.

The second problem set consists of **SOLAR** library instances [6], which possess explicit linear inequality constraints. This solar thermal power plant simulation is available at <https://github.com/bbopt/solar>. This set of problems is chosen since it has realistic characteristics of blackbox optimization. Three of the nine instances available in **SOLAR** are used. These problems have a single objective function to optimize and a priori linear inequality constraints mixed with unknown constraints. Some variables are discrete and are handled by the mesh in **NOMAD**. The Segment Search always projects trial points on the mesh. This means that these variables are well handled by the proposed strategies. Furthermore, if a point $Proj(t)$ does not respect the unknown constraints, it is handled by the extreme barrier.

SOLAR has stochastic outputs that vary according to random seeds and accepts a fidelity value in $]0, 1]$. A given fidelity corresponds to a surrogate more or less true to the actual

Table 4.1 The provided set of starting points with their associated function value for each selected problem of the Hock-Schittkowski problem collection.

Problem	Starting point (x_0)	$f(x_0)$
HS21	(10, 10)	1
HS24	(1, 0.5)	-0.0133646
HS35	(0.5, 0.5, 0.5)	2.25
HS36	(10, 10, 10)	-1000
HS37	(10, 10, 10)	-1000
HS44	(0, 0, 0, 0)	0
HS76	(0.5, 0.5, 0.5, 0.5)	-1.25
HS86	(0, 0, 0, 0, 1)	20.0
HS105	(0.1, 0.2, 100, 125, 175, 11.2, 13.2, 15.8)	1608.38

blackbox function. For smaller fidelity values, the runtime of an evaluation is smaller, and values returned by the blackbox are less true to the original problem. For the tests, the fidelity of **SOLAR** 1 is set to 1. **SOLAR** 2 and 7 are set to 0.05. For each of the three problems, five different seeds are used.

Table 4.2 The provided starting points of the **SOLAR** problems where the fidelity is set to 0.05 and where the i in f_i denotes the seed index.

Problem	Starting point x_0	$f_1(x_0)$	$f_2(x_0)$	$f_3(x_0)$	$f_4(x_0)$	$f_5(x_0)$
SOLAR #1, fid=1	(8, 8, 150, 7, 7, 250, 45, 0.5, 5)	-116441 .966918	-118093 .801195	-122434. 553591	-116441. 966918	-118097. 65666
SOLAR #2, fid=0.05	(11.0, 11.0, 140.0, 10.0, 10.0, 2650, 89.0, 0.5, 5.0, 838.0, 36, 0.30, 0.020, 0.0216)	753526. 705927	753526. 705927	753526. 705927	753526. 705927	753526. 705927
SOLAR #7, fid=0.05	(7.0, 7.0, 850.0, 40, 0.20, 0.010, 0.0110)	-56.75 95418979	-56.74 76593411	-56.80 41797834	-56.75 78619552	-56.74 14335665

Finding points in $\mathcal{X}$ for the **SOLAR** problems is explained in Section 4.3.

Finally, **NOMAD** accepts seeds for the generation of the polling directions. Five seeds are chosen for each problem (Hock-Schittkowski and **SOLAR**). Selected seeds for **NOMAD** are displayed in Appendix A.1.

The instances for the two different test sets are defined as follows.

DEFINITION 4.1: INSTANCE DEFINITION FOR THE HOCK-SCHITTKOWSKI PROBLEM COLLECTION

A problem **instance** for the Hock-Schittkowski problem collection consist of the **analytical model** associated with a **starting point** and a running **NOMAD seed**

DEFINITION 4.2: INSTANCE DEFINITION FOR SOLAR PROBLEM

A problem **instance** for a SOLAR simulation consists of the simulation problem associated with a **starting point**, a running **SOLAR seed**, a running **NOMAD seed** and a **fidelity** value.

4.2 Data Profiles

Data profiles are valuable tools designed to evaluate the performance of algorithms. In the article [35] by Moré and Wild, they are defined for unconstrained optimization problems. They are the tools that will be used to compare the different proposed strategies described in Section 3. In this Section, they are adapted to handle the constrained test set of this work. The following list presents notation in order to define them. Most of the notation is taken from the article by Moré and Wild.

- $\mathcal{S}$ a set of optimization solvers with elements $s \in \mathcal{S}$.
- $\mathcal{P}$ a set of problems with elements $p \in \mathcal{P}$.
- $t_{p,s} \in \mathbb{R}$ a performance measure associated with problem p and solver s .
- $\alpha \in \mathbb{R}$ a threshold on the performance measure.
- $\tau \in \mathbb{R} : \tau \in]0, 1[$ a tolerance.
- $n_p \in \mathbb{N}$ the dimension of problem $p \in \mathcal{P}$ and $\mathcal{N}$ the set of all the n_p .
- $n_{p_{\min}} \in \mathbb{N}$ the smallest dimension $n_p \in \mathcal{N}$.
- $e \in \mathbb{Z} : 0 \leq e \leq b$ an evaluation limited by the allocated budget of evaluations b .

For data profiles, the performance measure $t_{p,s}$ used is the smallest number of blackbox evaluations $e_{p,s}$ required for solver s applied to problem p to satisfy a convergence condition. Many

convergence conditions can be defined. The convergence condition used for the experiments is detailed below in 4.3.

DEFINITION 4.3: THE CONVERGENCE CONDITION TO SATISFY TAKEN FROM [35]

$$\frac{f(x_e) - f(x_0)}{f_L - f(x_0)} \geq (1 - \tau), \quad (4.1)$$

where τ , the tolerance is generally smaller than 10^{-1} .

Generally, f_L can be a solution that is better or equal to the best solution found by the solvers. For this experimental design, f_L is the best solution for problem p over all the solvers $s \in \mathcal{S}$ as defined in [35]. Therefore, the convergence condition (4.3) verifies if the gap $f(x_e) - f(x_0)$ is close enough to the initial gap between the value of f at the starting point x_0 and the best solution f_L . If these two gaps are close enough, the solver s was able to find a good solution relative to the starting point x_0 .

Condition (4.3) is defined for unconstrained optimization. However, in constrained optimization, the value of $f(x)$ outside of Ω does not have a meaning. In that case, a value must be assigned for $f(x)$ when $x \notin \Omega$. It is done by the extreme barrier function f_Ω .

DEFINITION 4.4: THE DATA PROFILE DEFINITION TAKEN FROM [35]

A data profile $d_s(\alpha)$ for solver $s \in \mathcal{S}$ over all the problems $\mathcal{P}$ is

$$d_s(\alpha) = \frac{1}{|\mathcal{P}|} |\{p \in \mathcal{P} \mid t_{p,s} \leq \alpha\}|, \quad (4.2)$$

where $\alpha \in [0, \frac{b}{n_{p_{\min}}+1}]$ is a number of simplex gradient estimates, and b is the budget of blackbox evaluations that was allowed for all the solvers $s \in \mathcal{S}$.

One simplex gradient estimate for a problem p is $n_p + 1$ where n_p is the dimension of problem p and is exactly defined in [35] as "the number of evaluations needed to compute a one-sided finite-difference estimate of the gradient." It also allows us to normalize the evaluations according to the dimension of the problems.

Concisely, the data profile $d_s(\alpha)$ defined over $\mathcal{P}$ and $\mathcal{S}$ represents, for a fixed value of simplex gradient estimates α , the ratio of problems solved by solver s . A problem is considered to be solved after $\frac{t_{p,s}}{n_p+1}$ simplex gradient estimates if it respects the convergence Condition (4.3) where a tolerance τ is fixed. For the pair $(\mathcal{P}, \mathcal{S})$, τ can vary to produce different data profiles.

4.3 Experiments

This Section details the test cases. The first experiment is running VBP, APEB, SC, and L2 on the Hock-Schittkowski problem instances with starting points inside $\mathcal{X}$. The strategies are compared using **NOMAD**, a C++MADS implementation. A basic version of MADS that does not use sophisticated searches is used. The strategies are tested with and without the Segment Search heuristic.

The second experiment runs VBP, APEB, SC, and L2 on the **SOLAR** problem instances. The starting points provided for the **SOLAR** problems are not necessarily in $\mathcal{X}$ or in Ω . If they are not, **NOMAD** has a first phase called Phase One that attempts to find a point in Ω by minimizing a cost function that includes the constraints that are not respected. The first phase is applied without using any of the strategies. When a point in Ω is found, the strategies are activated. This phase is necessary because the strategies assume that x^k is in $\mathcal{X}$. Since starting points outside of $\mathcal{X}$ could happen in practice, it is an interesting case to test the strategies with this approach. For a given **NOMAD** seed and a given **SOLAR** seed, Phase One will reach the same starting point inside Ω . Therefore, the first phase introduces the same lag in the number of evaluations for the SC, L2, and APEB strategies.

Then, the Quadratic Search option of **NOMAD** is turned on, and the strategies are tested on the Hock-Schittkowski problems. The Quadratic Search is based on [36]. It is a model-based search that proposes trial points using a quadratic approximation of the problem. However, if the quadratic model does not identify the linear inequality constraints, it can still propose trial points outside of $\mathcal{X}$. A simple modification is made to avoid this and consists of rejecting trial points if they are outside of $\mathcal{X}$ in the same way as in the APEB strategy. This modification is made for APEB, SC, and L2.

Finally, it is important to note that, by default, **NOMAD** does not use the mesh size parameter update as seen in 2 but a more sophisticated one. It is based on [37]. There are also many parameters to set that are not explicit in the framework. As stated previously, for the first test cases, there are no searches done in the search step of MADS other than the proposed strategies except for the quadratic case. The set of polling directions is of order $2n$ where n is the dimension of the problem. These polling directions are orthogonal and defined in [38]. Figure 4.1 shows an example of a parameter file generated for the first test case.

```

DIMENSION 2
BB_EXE HS21_KLIC.exe
BB_OUTPUT_TYPE OBJ
X0 ( 40.5163 -46.2458 )
MAX_BB_EVAL 1000
DISPLAY_DEGREE 2
DISPLAY_STATS SOL BBE OBJ
DISPLAY_INFEASIBLE yes
STATS_FILE "HS21-SEEDPOSITION1-SP5-1K_NOMAD-ProjectToConstraint.txt"
HISTORY_FILE "HIST-HS21-SEEDPOSITION1-SP5-1K_NOMAD-ProjectToConstraint.txt"
KNOWN_LINEAR_CONSTRAINT_STRATEGY ProjectToConstraint
NM_SEARCH false
QUAD_MODEL_SEARCH false
SPECULATIVE_SEARCH false
EVAL_QUEUE_SORT LEXICOGRAPHICAL
DIRECTION_TYPE ORTHO 2N |
SEED 1
LINEAR_CONSTRAINT_DEFINITION -10 1 -10
LINEAR_CONSTRAINT_DEFINITION -1 0 -2
LINEAR_CONSTRAINT_DEFINITION 1 0 50
LINEAR_CONSTRAINT_DEFINITION 0 -1 50
LINEAR_CONSTRAINT_DEFINITION 0 1 50

```

Figure 4.1 A parameter file for NOMAD where the search strategies are deactivated, and the number of polling directions is set to $2n$.

CHAPTER 5 EXPERIMENTAL RESULTS

5.1 T-Searches and VBP on the Hock-Schittkowski problems

The first data profiles are presented in Figures 5.1 and 5.2. They compare APEB, SC, L2, and VBP without the Segment Search mesh projection on the Hock-Schittkowski problems. It is important to note that the APEB and VBP strategies do not require a mesh projection by definition.

The budget allowed for any problem is 1000 evaluations. For each problem, the best solution used for the convergence criteria is the best solution found across all the strategies as defined in Section 4.2.

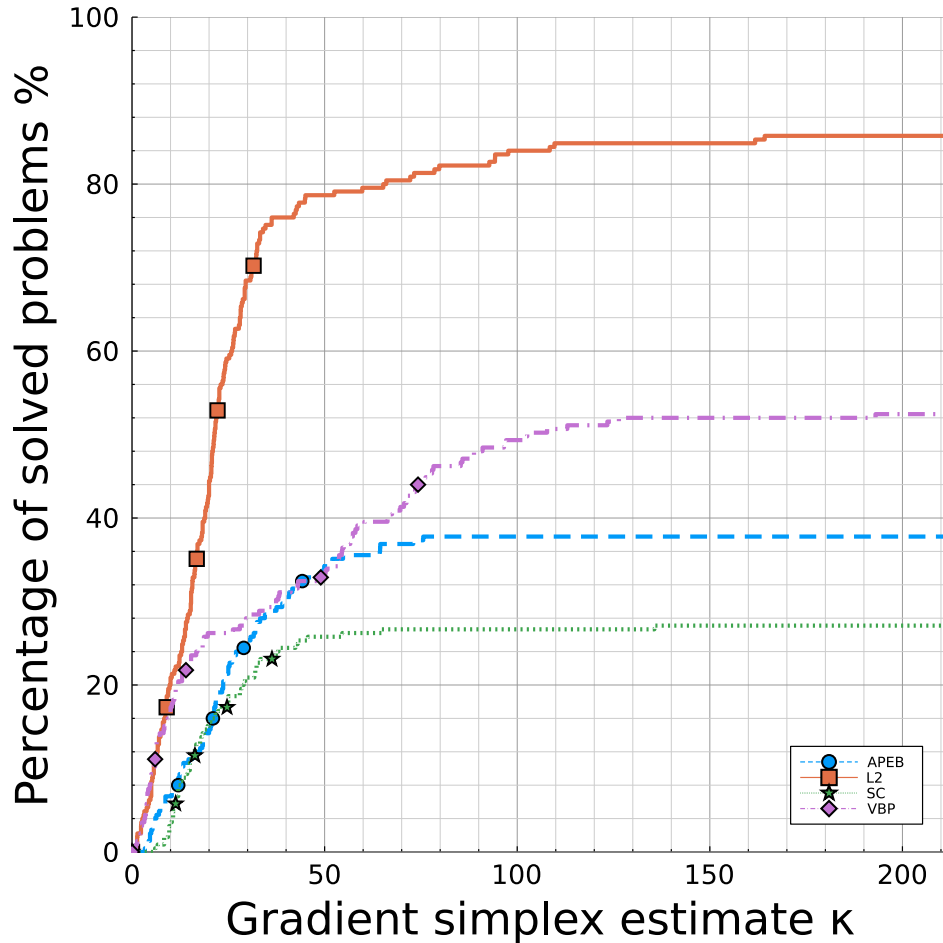


Figure 5.1 Data profile of the strategies without Segment Search with tolerance 1E-3 using 1E3 evaluations on the Hock-Schittkowski problems.

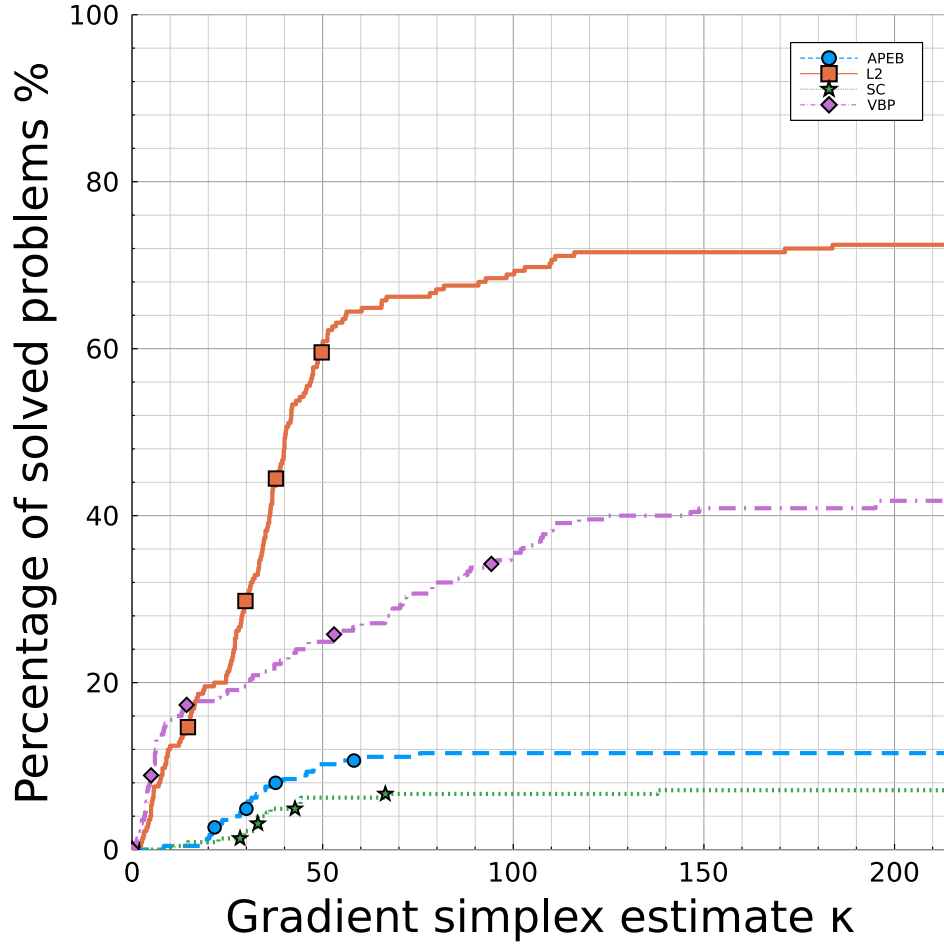


Figure 5.2 Data profile of the strategies without Segment Search with tolerance $1E-7$ using $1E3$ evaluations on the Hock-Schittkowski problems.

The data profiles in Figures 5.1 and 5.2 clearly indicate that without the Segment Search, L2 (the ℓ_2 Norm Projection) finds better solutions overall. For $\tau = 1E-7$, after 1000 evaluations, L2 solves $\approx 72\%$ of the problems according to the convergence criteria, while VBP is the second best strategy with $\approx 42\%$ solved problems for the same budget. For $\tau = 1E-3$, after 1000 evaluations, L2 solves $\approx 86\%$ of the problems according to the convergence criteria, while VBP is the second best strategy with $\approx 52\%$ solved problems for the same budget. The SC strategy is the worst, and all the other strategies dominate it.

The curves also show that, for $\tau = 1E-7$, after ≈ 20 gradient simplex estimates, the number of problems solved by L2 is always greater than the other strategies. For $\tau = 1E-3$, after ≈ 10 gradient simplex estimates, the number of problems solved by L2 is always greater than the other strategies.

The fact that the MADS algorithm converges if the trial points are projected on the mesh does not necessarily mean that the MADS algorithm without mesh projection does not converge for the proposed strategies. However, since there is no proof of this, the adopted strategies will necessarily use the Segment Search.

The data profile presented in Figure 5.3 compares the different strategies where L2 and SC use the Segment Search mesh projection. The number of allowed blackbox evaluations is 1000, and the tolerances used for the convergence criteria are $1E-3$ and $1E-7$.

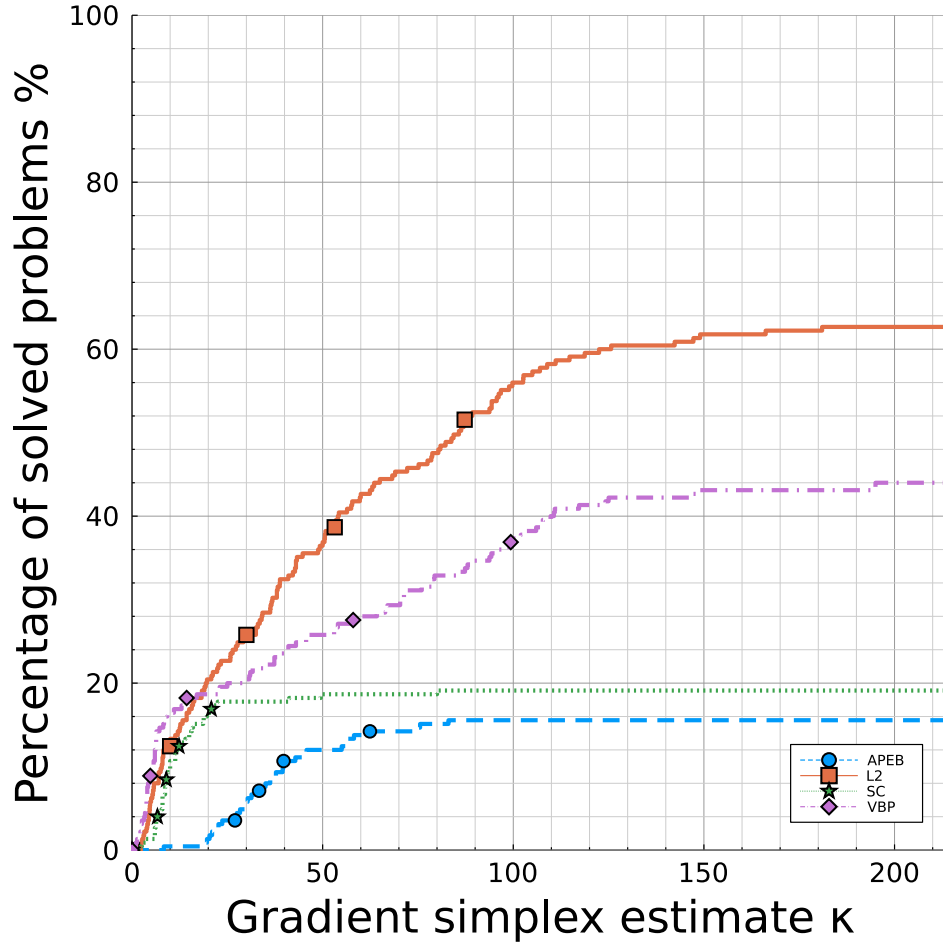


Figure 5.3 Data profile of the strategies with tolerance $1E-7$ using $1E3$ evaluations on the Hock-Schittkowski problems.

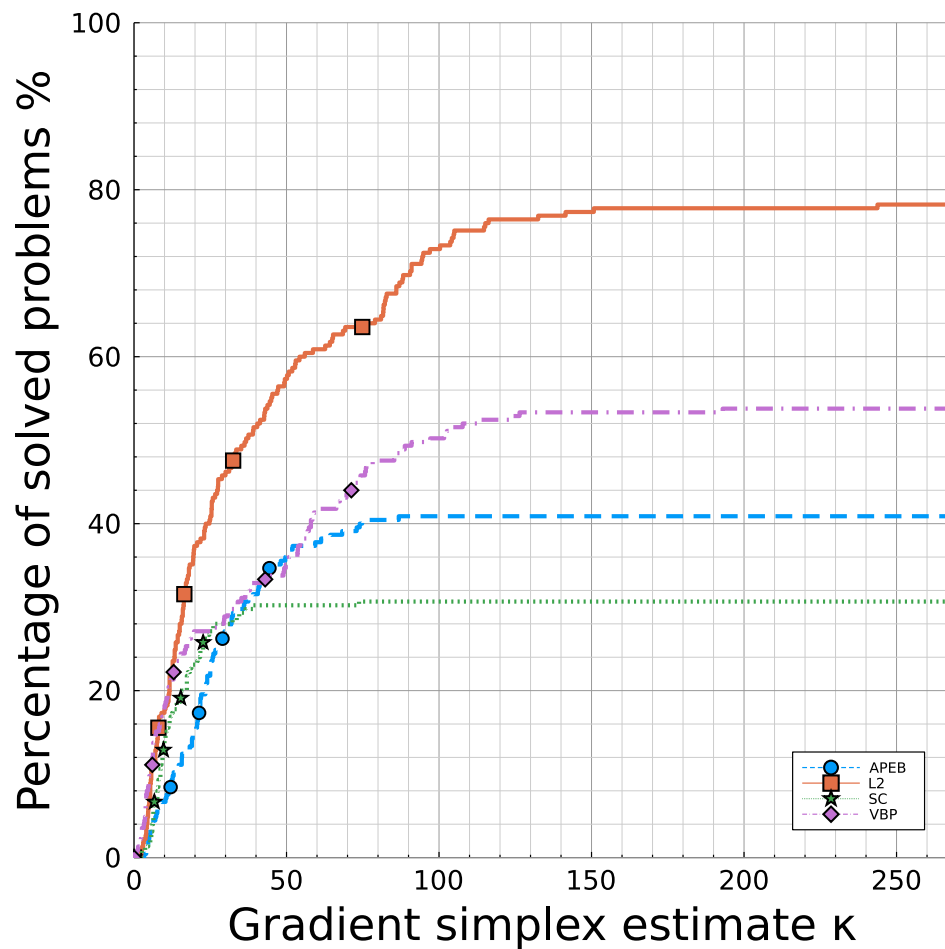


Figure 5.4 Data profile of the strategies with tolerance $1E-3$ using $1E3$ evaluations on the Hock-Schittkowski problems.

Figures 5.3 and 5.4 show again that L2 is the best strategy for this set of problems.

Figures 5.5 and 5.6 compare, for tolerance $1E-7$ and $1E-3$, respectively, the strategies that use the Segment Search against the strategies devoid of mesh projections using the same parameters as before.

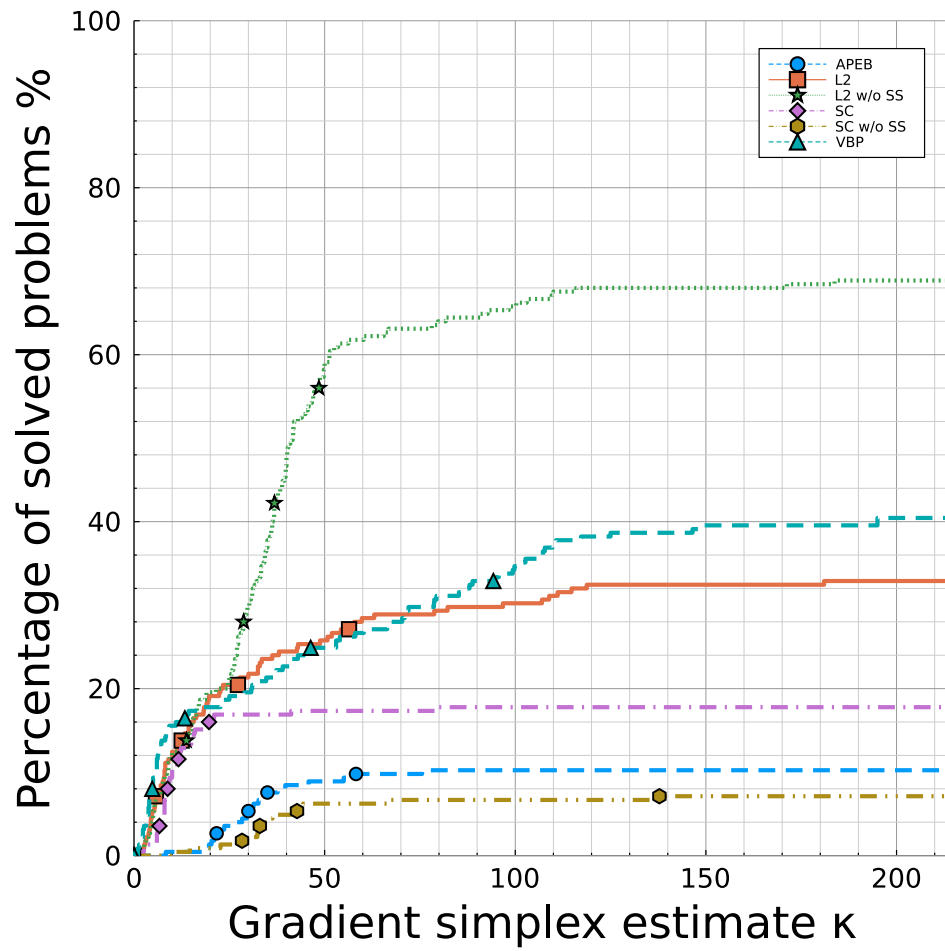


Figure 5.5 Data profile of the strategies with and without Segment Search with tolerance $1E-7$ using $1E3$ evaluations on the Hock-Schittkowski problems.

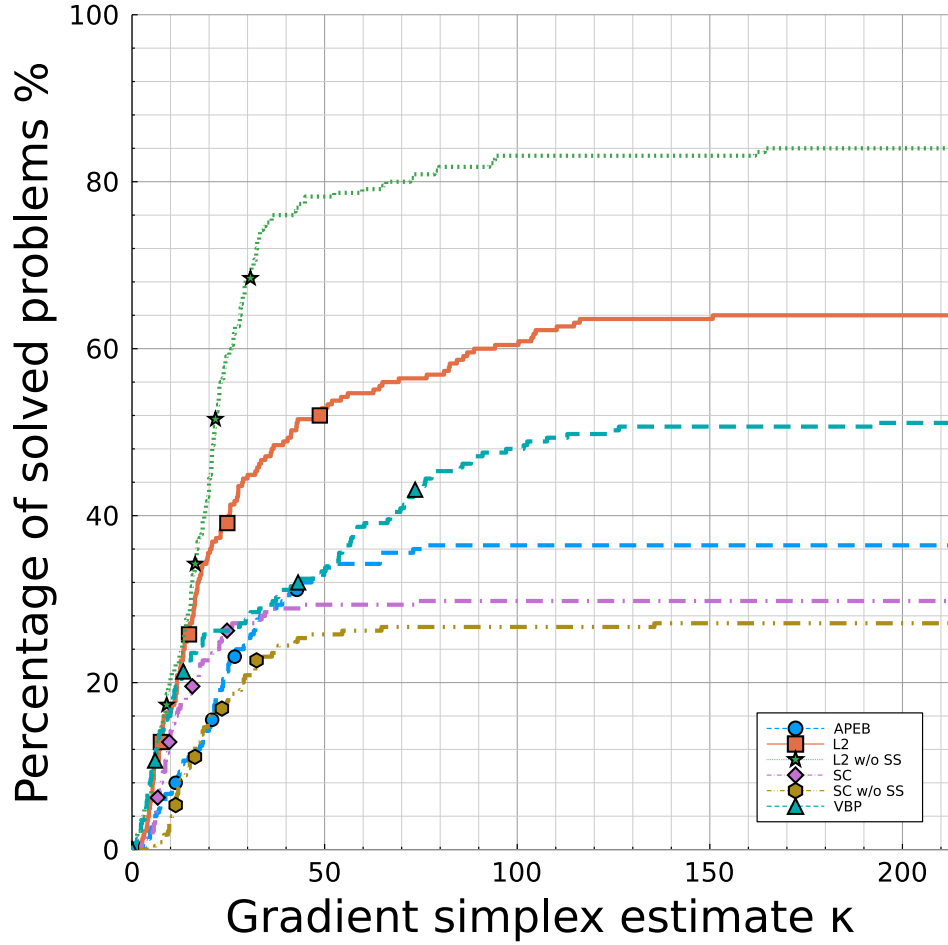


Figure 5.6 Data profile of the strategies with and without Segment Search with tolerance $1E-3$ using $1E3$ evaluations on the Hock-Schittkowski problems.

For both tolerances, the data profiles show that the L2 strategy without SS is the best on this set of problems. It dominates all of the other ones after approximately 25 gradient simplex estimates. The L2 without SS does not satisfy the MADS framework and lacks a convergence proof. Thus the next best strategy must be identified. With a small tolerance $\tau = 1E-7$, the second choice is the VBP, and with tolerance $\tau = 1E-3$, it is L2. Furthermore, the SC strategy seems to perform better with the Segment Search. It dominates the APEB strategy for $\tau = 1E-7$ and consistently dominates the SC strategy without SS.

5.2 T-Searches and VBP on the SOLAR problems

Figures 5.7 and 5.8 compare APEB, SC, L2, and VBP on the SOLAR problems. Table 4.2 shows that the value of f for different SOLAR seeds is the same. After numerical investigation, the values of f and c are the same for different seeds in the SOLAR 2 problem. Consequently, the different seeds for SOLAR 2 are not included in this section's data profiles. Only the first seed of SOLAR 2 is kept.

The budget allowed for any problem is 1000 evaluations. For each problem, the best solution used for the convergence criteria is the best solution found across all the strategies as defined in Section 4.2.

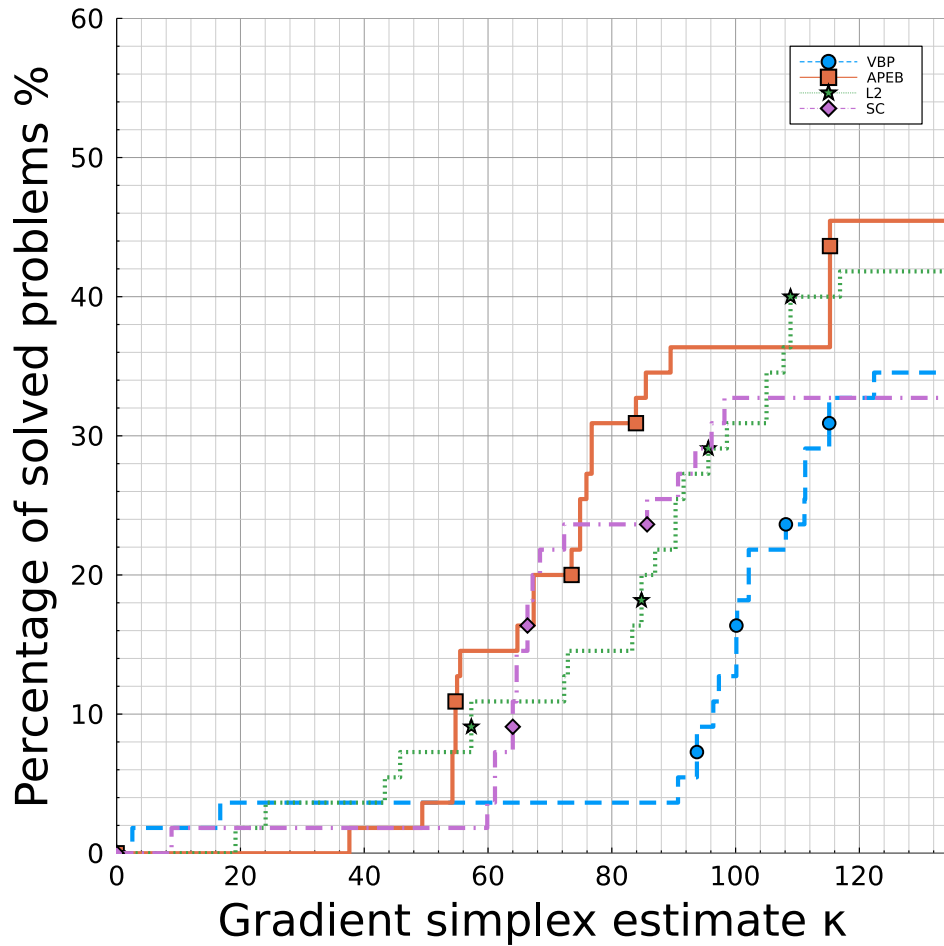


Figure 5.7 Data profile of the strategies with tolerance 1E-3 using 1E3 evaluations on the SOLAR problems.

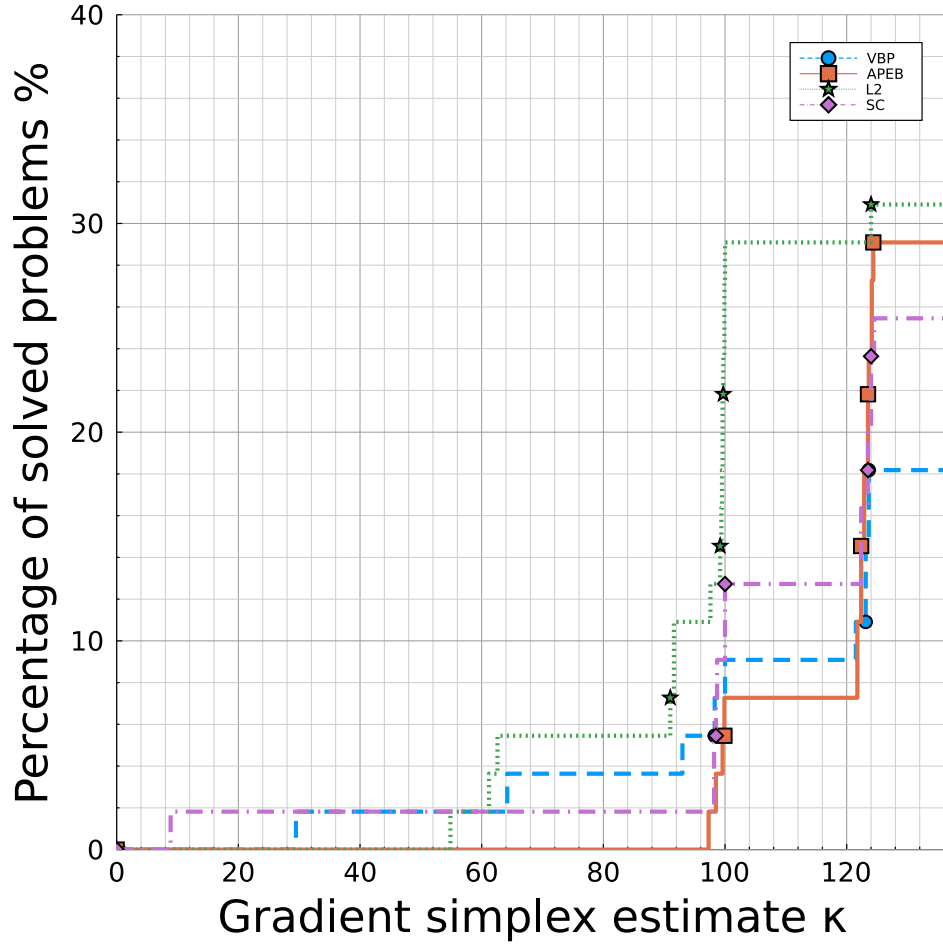


Figure 5.8 Data profile of the strategies with tolerance $1\text{E-}7$ using $1\text{E}3$ evaluations on the SOLAR problems.

For $\tau = 1\text{E-}3$, Figure 5.7 shows that for small number of gradient simplex estimates, the L2 strategy solves fewer problems than APEB and SC. After 1000 blackbox evaluations, L2 solves $\approx 42\%$ of the problems, and APEB solves $\approx 46\%$ of them. For $\tau = 1\text{E-}7$, Figure 5.8 shows that after the budget of evaluations is spent L2, APEB, SC, and VBP respectively solve $\approx 31\%$, $\approx 29\%$, $\approx 25\%$ and $\approx 18\%$ of the problems. For the SOLAR problems, the worst strategy is VBP, and the best is L2 followed closely by APEB.

5.3 The Quadratic Search with T-Searches and VBP on the Hock-Schittkowski problems

The Quadratic Search option in NOMAD is based on [36]. It is a powerful option with strong performance. Thus, it is interesting to look at the behavior of the strategies mixed with it. Figure 5.9 and 5.10 show the different strategies mixed with the Quadratic Search for $\tau = 1E-3$ and $\tau = 1E-7$. For the T-Search strategies, when the trial points proposed by the Quadratic Search are outside of $\mathcal{X}$, they are not evaluated.

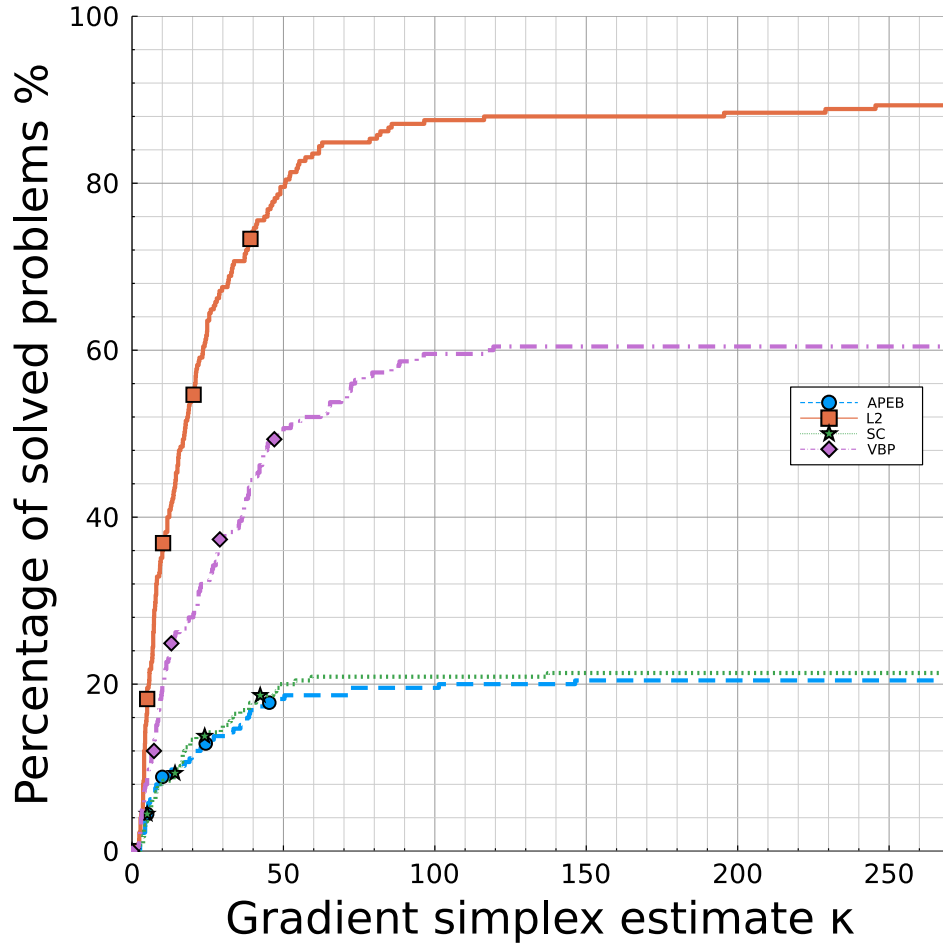


Figure 5.9 Data profile of the strategies mixed with the Quadratic Search with tolerance $1E-3$ using $1E3$ evaluations on the Hock-Schittkowski problems.

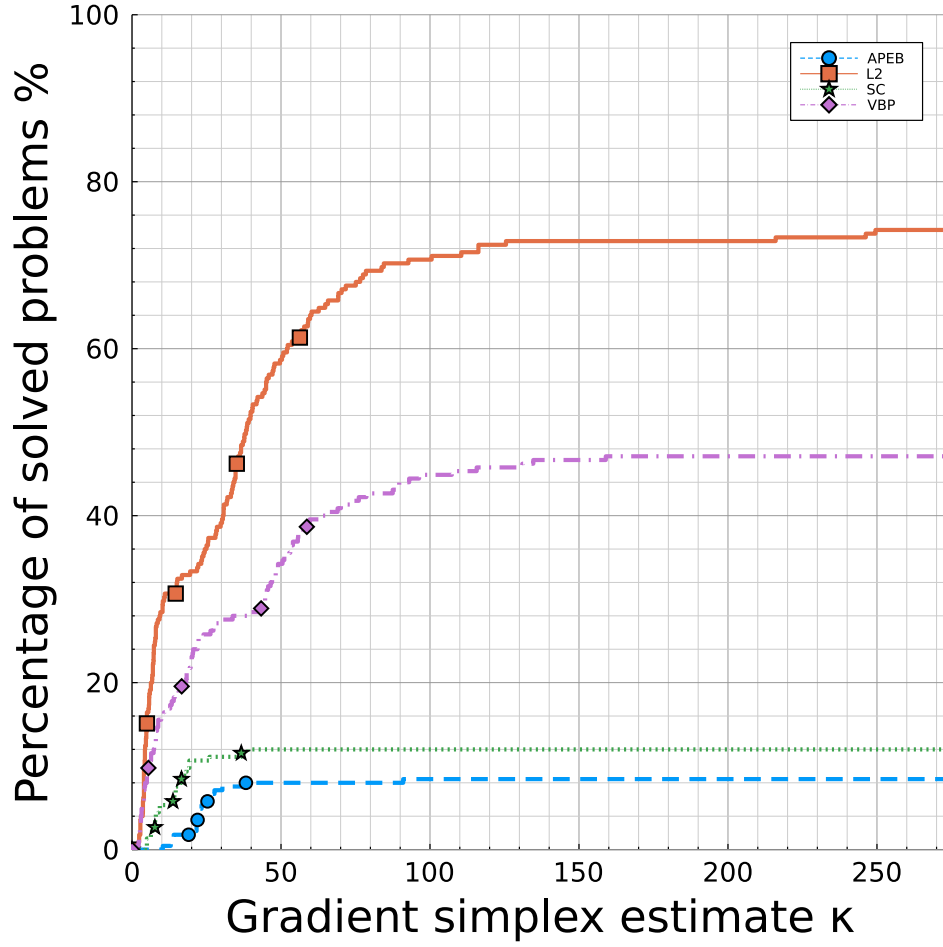


Figure 5.10 Data profile of the strategies mixed with the Quadratic Search with tolerance $1\text{E-}7$ using $1\text{E}3$ evaluations on the Hock-Schittkowski problems.

Figures 5.9 and 5.10 strongly suggests that the L2 strategy mixed with the Quadratic Search is a powerful combination. For $\tau=1\text{E-}3$, L2 solves $\approx 90\%$ of the problems after 1000 blackbox evaluations. The second best strategy for the same number of evaluations is VBP with $\approx 60\%$ of solved problems. Both QUAD APEB and QUAD SC underperform with $\approx 21\%$ problem solved. For $\tau=1\text{E-}7$, QUAD L2 solves $\approx 74\%$ after 1000 blackbox evaluations. The second best strategy is VBP with only $\approx 47\%$ solved problems.

5.4 Recommendation

Across the different data profiles, the L2 strategy had the best performance. Except for the **SOLAR** problems with $\tau = 1\text{E-}3$ and the mixed data profile of the Hock-Schittkowski problems for $\tau = 1\text{E-}7$, L2 was able to solve more problems for the given budget of evaluations. Its transient behavior was always dominating the other strategies after a relatively small number of gradient simplex estimates except for the **SOLAR** problems with $\tau = 1\text{E-}3$ and the mixed data profile of the Hock-Schittkowski problems with $\tau = 1\text{E-}7$. VBP performs well for the analytical problems of the Hock-Schittkowski problem collection but not for the more practical problems of **SOLAR**. Hence, the L2 strategy should be the default T-Search used in the **NOMAD** software and in the MADS framework when explicit linear inequality constraints are provided.

CHAPTER 6 SOFTWARE DESIGN CHOICES

In this chapter, details and choices for implementing the strategies in **NOMAD** will be covered.

6.1 Providing linear inequality constraints to **NOMAD**

Running **NOMAD** on a blackbox problem is done by providing a parameter file that contains information on the problem, such as its dimension, a starting point, and the blackbox executable that outputs $c(x)$ and $f(x)$. One can also indicate upper and lower bounds for a variable x_i . This parameter file also contains information about how to treat each blackbox constraint. The acronym **EB** must be indicated for unrelaxable constraints, while the acronym **PB** tells **NOMAD** to use the progressive barrier approach for relaxable constraints.

User-wise, two things had to be considered to integrate the strategies in **NOMAD**. The first is the way of indicating that some of the constraints are known explicitly. The second is the way of providing the linear inequality constraints to **NOMAD**.

The linear inequality constraints are provided to **NOMAD** with the following nomenclature in the parameter file.

How to provide a linear inequality to **NOMAD**

`LINEAR_CONSTRAINT_DEFINITION` $a_{i,0} \ a_{i,1} \ \dots \ a_{i,n-1} \ a_{i,n} \ b_i$,

where i represents a line, n is the dimension of the problem and a space separates each element.

A chosen strategy is provided in the following way.

How to provide the strategy to use to **NOMAD**

`KNOWN_LINEAR_CONSTRAINT_STRATEGY` *strategy*,

where *strategy* is the name of the strategy.

If some outputs of the blackbox executable are the linear inequality constraints, they should be identified with **EB**. Since the strategies always provide trial points inside $\mathcal{X}$, setting **EB** for these constraints will not influence **NOMAD** because the extreme barrier will never be used.

If one or more `LINEAR_CONSTRAINT_DEFINITION` is defined, then **NOMAD** will automatically use **L2** by default. If the blackbox does not return these constraints, they will still

be considered. The current implementation does not consider the upper and lower bounds on the variables when using the linear inequality constraints strategies. Consequently, they must be provided via the linear inequality constraint definitions. For a user-friendly interface, the upper and lower bounds should be converted to linear inequality constraints in a subsequent implementation.

Figure 6.1 shows an example of this nomenclature in a parameter file for HS21.

```

DIMENSION 2
BB_EXE HS21_KLIC.exe
BB_OUTPUT_TYPE OBJ EB EB EB EB EB
X0 ( 7.6389 17.1388 )|
MAX_BB_EVAL 1000

KNOWN_LINEAR_CONSTRAINT_STRATEGY SnapToConstraint

LINEAR_CONSTRAINT_DEFINITION -10 1 -10
LINEAR_CONSTRAINT_DEFINITION -1 0 -2
LINEAR_CONSTRAINT_DEFINITION 1 0 50
LINEAR_CONSTRAINT_DEFINITION 0 -1 50
LINEAR_CONSTRAINT_DEFINITION 0 1 50

```

Figure 6.1 A parameter file for **NOMAD** where the explicit linear inequality constraints are specified and where the chosen strategy is Snap-To-Constraints.

6.2 Solving the strategies' subproblems

The first subproblem to solve arises in the ℓ_2 projection. It is not trivial. However, it can be formulated as a quadratic program, and many solvers can be used to solve it. For the experiments, the Gurobi solver was integrated into **NOMAD** in order to find the subproblem solution at each iteration. Furthermore, for the L2 strategy, the result of the euclidian norm projection for different points can be the same. A safety net must be implemented to avoid multiple evaluations of the same point.

Subproblem 3.2 of the SC strategy can be computed in two different ways. At each iteration, the lines of the system $Ax \leq b$ are tested at trial point t . The index of each line that is not respected is kept in memory. The calculations (3.6) are only done for these unrespected constraints. Since each line $A_i x \leq b_i$ of the system $Ax \leq b$ represents a convex set, any point in the segment $L(s)$ will also reside in $A_i x \leq b_i$ and the segment will never intersect the hyperplane i by definition of a convex set. Another option is to check all the lines until one is not respected, stop, and compute all of the s_i .

CHAPTER 7 CONCLUSION AND RECOMMENDATIONS

7.1 Summary of works

This work proposes a modified MADS framework to efficiently handle explicit linear inequality constraints. Two classes of search, denoted T-Searches and S-Searches, are introduced as part of the MADS framework to exploit $\mathcal{X}$. Three T-Searches are proposed: the Snap-To-Constraints (SC), the ℓ_2 Norm Projection (L2), and the a priori extreme barrier (APEB). The L2 and SC strategies use a mesh projection heuristic called the Segment Search to inherit the convergence properties of MADS. Another strategy that does not use the modified framework was considered, the VBP. VBP consists of finding the variables' bounds a priori by solving linear programming problems and directly providing them to **NOMAD** for treatment. A preliminary version of these strategies is implemented in the **NOMAD** software.

Experiments involving the previous strategies and hybrid searches using the Quadratic Search shows that the most promising strategy was L2.

7.2 Limitations

This work's set of practical problems is restricted to only three **SOLAR** problems. Furthermore, the proposed strategies were only mixed with the Quadratic Search. They should be mixed with other state-of-the-art searches such as the Nelder-Mead Search [39] and the Speculative Search [18] to definitively conclude that they bring an edge.

The blackbox evaluations budget was only 1000 for all the tests. In practice, there could exist problems where more than 1000 evaluations are allowed. Therefore, the tests should be run again with more than 1000 evaluations to ensure no strategies get better with a greater budget.

The VBP strategy evaluates trial points outside of $\mathcal{X}$. Therefore, it could perform better than L2 if it excludes them. An experiment should be designed to verify this.

7.3 Future research

Future work could develop new strategies to handle linear inequality constraints using Algorithm 3. In this work, only T-Searches were designed. One could test different T-Searches by defining new projections. The ℓ_1 projection could be investigated. Since the ℓ_1 norm projection can be formulated as a linear programming problem, it could be solved faster than the ℓ_2 norm projection problem. Furthermore, already existing search step implementations such as the Nelder-Mead one [39] could be redesigned to propose trial points in $\mathcal{X}$. Such implementations would be classified as S-Searches.

An interesting research avenue would be to use the Segment Search directly on general inequality constraints by taking uniformly separated points along the segment $L(s)$. The ℓ_2 norm strategy could be implemented on general convex constraints by adding these constraints to the subproblem. Relaxable constraints could also be considered in the design of strategies.

For practical problems such as SOLAR, methods that use $\mathcal{X}$ could be designed to find starting points in Ω using fewer blackbox evaluations.

The convergence of the L2 strategy without SS should be investigated. If the convergence properties of MADS stay valid, L2 without SS could be the best, as suggested by the data profile in Figure 5.5 for nongranular input. In this case, it should be the default implementation to use.

A comparison of the different methods available in the literature that deal with explicit linear inequality constraints would provide further insight into the quality of the proposed strategies.

REFERENCES

- [1] C. Audet and W. Hare, *Derivative-Free and Blackbox Optimization*, ser. Springer Series in Operations Research and Financial Engineering. Cham, Switzerland: Springer, 2017. [Online]. Available: <https://dx.doi.org/10.1007/978-3-319-68913-5>
- [2] J. Lainé, E. Piollet, F. Nyssen, and A. Batailly, “Blackbox Optimization for Aircraft Engine Blades With Contact Interfaces,” *Journal of Engineering for Gas Turbines and Power*, vol. 141, no. 6, 2019. [Online]. Available: <https://dx.doi.org/10.1115/1.4042808>
- [3] C. Audet, A. Batailly, and S. Kojtych, “Escaping unknown discontinuous regions in blackbox optimization,” *SIAM Journal on Optimization*, vol. 32, no. 3, pp. 1843–1870, 2022. [Online]. Available: <https://doi.org/10.1137/21M1420915>
- [4] S. Alarie, C. Audet, V. Garnier, S. Le Digabel, and L.-A. Leclaire, “Snow water equivalent estimation using blackbox optimization,” *Pacific Journal of Optimization*, vol. 9, no. 1, pp. 1–21, 2013. [Online]. Available: <http://www.ybook.co.jp/online2/oppjo/vol9/p1.html>
- [5] S. Alarie, C. Audet, A. E. Gheribi, M. Kokkolaras, and S. Le Digabel, “Two decades of blackbox optimization applications,” *EURO Journal on Computational Optimization*, vol. 9, p. 100011, 2021. [Online]. Available: <https://doi.org/10.1016/j.ejco.2021.100011>
- [6] M. Lemyre Garneau, “Modelling of a solar thermal power plant for benchmarking blackbox optimization solvers,” Master’s thesis, Polytechnique Montréal, 2015. [Online]. Available: <https://publications.polymtl.ca/1996/>
- [7] C. Audet, P. Côté, C. Poissant, and C. Tribes, “Monotonic grey box optimization,” *Optimization Letters*, vol. 14, no. 1, pp. 3–18, 2020. [Online]. Available: <https://dx.doi.org/10.1007/s11590-019-01497-8>
- [8] C. Audet, V. Béchar, and S. Le Digabel, “Nonsmooth optimization through Mesh Adaptive Direct Search and Variable Neighborhood Search,” *Journal of Global Optimization*, vol. 41, no. 2, pp. 299–318, 2008. [Online]. Available: <https://dx.doi.org/Doi:10.1007/s10898-007-9234-1>
- [9] R. M. Lewis, V. Torczon, and M. W. Trosset, “Direct search methods: then and now,” *Journal of Computational and Applied Mathematics*, vol. 124, no. 1, pp. 191–207, 2000,

- numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0377042700004234>
- [10] A. R. Conn, K. Scheinberg, and L. N. Vicente, *Introduction to Derivative-Free Optimization*, ser. MOS-SIAM Series on Optimization. Philadelphia: SIAM, 2009. [Online]. Available: <https://dx.doi.org/10.1137/1.9780898718768>
 - [11] J. A. Nelder and R. Mead, “A Simplex Method for Function Minimization,” *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965. [Online]. Available: <http://comjnl.oxfordJournals.org/content/7/4/308.abstract>
 - [12] A. I. F. Vaz and L. N. Vicente, “PSwarm: a hybrid solver for linearly constrained global derivative-free optimization,” *Optimization Methods and Software*, vol. 24, no. 4-5, pp. 669–685, 2009. [Online]. Available: <https://dx.doi.org/10.1080/10556780902909948>
 - [13] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, “Optimization by Simulated Annealing,” *Science*, vol. 220, no. 4598, pp. 671–680, 1983. [Online]. Available: <http://www.jstor.org/stable/1690046>
 - [14] T. G. Kolda, R. M. Lewis, and V. Torczon, “Optimization by direct search: New perspectives on some classical and modern methods,” *SIAM Review*, vol. 45, no. 3, pp. 385–482, 2003. [Online]. Available: <https://dx.doi.org/Doi:10.1137/S003614450242889>
 - [15] V. Torczon, “On the convergence of pattern search algorithms,” *SIAM Journal on Optimization*, vol. 7, no. 1, pp. 1–25, 1997. [Online]. Available: <https://dx.doi.org/10.1137/S1052623493250780>
 - [16] R. M. Lewis and V. Torczon, “Pattern search methods for linearly constrained minimization,” *SIAM Journal on Optimization*, vol. 10, no. 3, pp. 917–941, 2000. [Online]. Available: <http://epubs.siam.org/sam-bin/dbq/article/33137>
 - [17] M. A. Abramson, O. A. Brezhneva, J. E. Dennis Jr., and R. L. Pingel, “Pattern search in the presence of degenerate linear constraints,” *Optimization Methods and Software*, vol. 23, no. 3, pp. 297–319, 2008. [Online]. Available: <https://dx.doi.org/10.1080/10556780701521670>
 - [18] C. Audet and J. E. Dennis Jr., “Mesh Adaptive Direct Search Algorithms for Constrained Optimization,” *SIAM Journal on Optimization*, vol. 17, no. 1, pp. 188–217, 2006. [Online]. Available: <https://dx.doi.org/Doi:10.1137/040603371>

- [19] F. H. Clarke, *Optimization and Nonsmooth Analysis*. New York: John Wiley and Sons, 1983, reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics. [Online]. Available: <http://www.ec-securehost.com/SIAM/CL05.html>
- [20] C. Audet, S. Le Digabel, V. Rochon Montplaisir, and C. Tribes, “Algorithm 1027: NOMAD version 4: Nonlinear optimization with the MADS algorithm,” *ACM Transactions on Mathematical Software*, vol. 48, no. 3, pp. 35:1–35:22, 2022. [Online]. Available: <https://dx.doi.org/10.1145/3544489>
- [21] C. Audet, S. Le Digabel, and M. Peyrega, “Linear equalities in blackbox optimization,” *Computational Optimization and Applications*, vol. 61, no. 1, pp. 1–23, 2015. [Online]. Available: <https://dx.doi.org/10.1007/s10589-014-9708-2>
- [22] Y. Zhang and L. Gao, “On numerical solution of the maximum volume ellipsoid problem,” *SIAM J. Optim.*, vol. 14, pp. 53–76, 2003. [Online]. Available: <https://doi.org/10.1137/S1052623401397230>
- [23] Y. Diouane, S. Gratton, and L. N. Vicente, “Globally convergent evolution strategies for constrained optimization,” *Computational Optimization and Applications*, vol. 62, no. 2, pp. 323–346, 2015. [Online]. Available: <https://dx.doi.org/10.1007/s10589-015-9747-3>
- [24] C. J. Price, M. Reale, and B. L. Robertson, “A cover partitioning method for bound constrained global optimization,” *Optimization Methods and Software*, vol. 27, no. 6, pp. 1059–1072, 2012. [Online]. Available: <https://doi.org/10.1080/10556788.2011.557726>
- [25] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang, “Direct search based on probabilistic feasible descent for bound and linearly constrained problems,” *Computational Optimization and Applications*, vol. 72, no. 3, pp. 525–559, Apr. 2019. [Online]. Available: <https://doi.org/10.1007/s10589-019-00062-4>
- [26] S. Gratton, C. Royer, L. N. Vicente, and Z. Zhang, “Direct search based on probabilistic descent,” *SIAM Journal on Optimization*, vol. 25, no. 3, pp. 1515–1541, 2015. [Online]. Available: <https://dx.doi.org/10.1137/140961602>
- [27] M. J. D. Powell, “On fast trust region methods for quadratic models with linear constraints,” *Mathematical Programming Computation*, vol. 7, no. 3, pp. 237–267, 2015. [Online]. Available: <https://dx.doi.org/10.1007/s12532-015-0084-4>
- [28] —, “Lincoa software,” Software available at <http://mat.uc.pt/~zhang/software.html#lincoa>. [Online]. Available: <http://mat.uc.pt/~zhang/software.html#lincoa>

- [29] —, “Least Frobenius norm updating of quadratic models that satisfy interpolation conditions,” *Mathematical Programming*, vol. 100, no. 1, pp. 183–215, 2004. [Online]. Available: <https://dx.doi.org/10.1007/s10107-003-0490-7>
- [30] E. A. E. Gumma, M. H. A. Hashim, and M. M. Ali, “A derivative-free algorithm for linearly constrained optimization problems,” *Computational Optimization and Applications*, vol. 57, no. 3, pp. 599–621, 2014. [Online]. Available: <https://dx.doi.org/10.1007/s10589-013-9607-y>
- [31] M. J. D. Powell, “The NEWUOA software for unconstrained optimization without derivatives,” in *Large-Scale Nonlinear Optimization*, ser. Nonconvex Optimization and Its Applications, P. Pardalos, G. Pillo, and M. Roma, Eds. Springer, 2006, vol. 83, pp. 255–297. [Online]. Available: https://dx.doi.org/10.1007/0-387-30065-1_16
- [32] A. Cristofari and F. Rinaldi, “A derivative-free method for structured optimization problems,” *SIAM Journal on Optimization*, vol. 31, no. 2, pp. 1079–1107, 2021. [Online]. Available: <https://doi.org/10.1137/20M1337417>
- [33] S. Le Digabel, “Algorithm 909: NOMAD: Nonlinear Optimization with the MADS algorithm,” *ACM Transactions on Mathematical Software*, vol. 37, no. 4, pp. 44:1–44:15, 2011. [Online]. Available: <https://dx.doi.org/10.1145/1916461.1916468>
- [34] W. Hock and K. Schittkowski, *Test Examples for Nonlinear Programming Codes*, ser. Lecture Notes in Economics and Mathematical Systems. Berlin, Germany: Springer, 1981, vol. 187.
- [35] J. J. Moré and S. M. Wild, “Benchmarking Derivative-Free Optimization Algorithms,” *SIAM Journal on Optimization*, vol. 20, no. 1, pp. 172–191, 2009. [Online]. Available: <https://dx.doi.org/10.1137/080724083>
- [36] A. R. Conn and S. Le Digabel, “Use of quadratic models with mesh-adaptive direct search for constrained black box optimization,” *Optimization Methods and Software*, vol. 28, no. 1, pp. 139–158, 2013. [Online]. Available: <https://dx.doi.org/10.1080/10556788.2011.623162>
- [37] C. Audet, S. Le Digabel, and C. Tribes, “The Mesh Adaptive Direct Search Algorithm for Granular and Discrete Variables,” *SIAM Journal on Optimization*, vol. 29, no. 2, pp. 1164–1189, 2019. [Online]. Available: <https://dx.doi.org/10.1137/18M1175872>
- [38] M. A. Abramson, C. Audet, J. E. Dennis Jr., and S. Le Digabel, “OrthoMADS: A Deterministic MADS Instance with Orthogonal Directions,” *SIAM Journal on*

- Optimization*, vol. 20, no. 2, pp. 948–966, 2009. [Online]. Available: <https://dx.doi.org/10.1137/080716980>
- [39] C. Audet and C. Tribes, “Mesh-based Nelder-Mead algorithm for inequality constrained optimization,” *Computational Optimization and Applications*, vol. 71, no. 2, pp. 331–352, 2018. [Online]. Available: <https://link.springer.com/article/10.1007/s10589-018-0016-0>

APPENDIX A GENERATING POINTS INSIDE $\mathcal{X}$

For the Hock-Schittkowski problems, the Latin Hypercube Sampling (LHS) defined below is used to generate starting points.

Algorithm 6: The Latin Hypercube Sampling (LHS) taken from [1].

Given a hyper-rectangle $[\ell, u] \subset \mathbb{R}^n$ and p the requested number of sample points.

0. Initialization

Let Π be a $n \times p$ matrix in which each of its n rows is a random permutation of the row vector $[1, 2, \dots, p]$.

1. Sample construction

Set $x_i^j = \ell_i + \frac{(\Pi_{i,j} - r_{i,j})}{p}(u_i - \ell_i)$ for each $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, p$,

where $r_{i,j} \in \mathbb{R}^n$ is randomly chosen in the interval $[0, 1)$.

Return $\{x^1, x^2, \dots, x^p\} \subset \mathbb{R}^n$.

A n -dimensional box is created according to the variables' bounds, and a Latin hypercube is used to sample this box. If one of the variables is unbounded, an arbitrary bound is set. The samples that are outside of the linear constraints are rejected. The bounds for a given variable are found with the simplex method on the linear constraints and forwarded to the LHS algorithm.

Tables A.1, A.2, A.3 and A.4 show the selected generated points #2, #3, #4 and #5 using the LHS method described above for each problem. The values of the starting points are truncated for readability.

Table A.1 The second set of starting points for the HS problems with their associated value.

Problem	Starting points (x_0) #2	$f(x_0)$
HS21	(24.93, 21.4839)	367.776
HS24	(0.199, 0.04817)	-2.76229e-06
HS35	(0.00244967, 0.054618, 0.58)	6.67815
HS36	(0.01633, 0.200267, 13.92)	-0.0455276
HS37	(0.0342954, 0.6554, 13.92)	-0.312899
HS44	(0.000786, 0.05546, 1.547499, 2.584)	-1.65885
HS76	(0.00051584, 0.04621739, 1.1606, 2.98797)	7.3129
HS86	(0.000754, 0.0717479, 0.21965, 0.1584, 0.72869)	-8.74661
HS105	(0.00143, 0.01024394, 130.95, 181.756, 238.406, 18.4845, 8.763889, 9.887)	2243.04

Table A.2 The third set of starting points for the HS problems with their associated value.

Problem	Starting points (x_0) #3	$f(x_0)$
HS21	(7.638895, 17.1387637)	194.321
HS24	(2.98, 1.578)	-0.756269
HS35	(2.64863, 0.0652, 0.04728)	1.86769
HS36	(5.1094, 9.40365, 10.966)	-526.895
HS37	(14.78, 17.12, 4.0166727)	-1016.49
HS44	(0.95095, 2.03394, 0.625, 2.05868)	-3.26064
HS76	(0.6534947, 0.17985, 0.851865, 0.5735)	0.350534
HS86	(0.1057, 0.24049, 0.2947, 0.39159, 0.52973)	-24.6576
HS105	(0.163986, 0.1930, 123.165, 131.592, 211.25, 15.02489, 24.8629, 11.279)	1293.1

Table A.3 The fourth set of starting points for the HS problems with their associated value.

Problem	Starting points (x_0) #4	$f(x_0)$
HS21	(8.478, 42.325)	1692.13
HS24	(4.413, 0.8075)	-0.0788498
HS35	(2.24875, 0.24547, 0.17364)	0.991989
HS36	(19.984, 4.4857, 1.65216687)	-148.106
HS37	(28.8, 15.46666, 0.5345458487)	-238.233
HS44	(2.04644, 0.5637393, 1.13994, 0.8569)	-0.0769003
HS76	(1.3469, 0.17, 0.89584, 2.10773)	2.46395
HS86	(0.2387, 0.2484, 0.299, 0.5661, 0.47327)	-26.0237
HS105	(0.376, 0.13314695, 111.61, 142.3719, 202.833, 19.5276, 11.874, 19.312329)	1254.88

Table A.4 The fifth set of starting points for the HS problems with their associated value.

Problem	Starting points (x_0) #5	$f(x_0)$
HS21	(40.516, -46.24584)	2055.09
HS24	(5.5738, 0.143468)	-0.000150006
HS35	(1.663, 0.0738, 0.530048)	0.964362
HS36	(15.95935, 9.1871988, 0.306)	-44.9158
HS37	(40.6849, 0.104124, 3.960)	-16.7765
HS44	(2.86, 0.1323, 0.7958, 1.57)	4.05445
HS76	(1.8325, 0.0652, 0.404, 2.4894554)	2.77467
HS86	(0.266, 0.057387, 0.35064, 0.8485363, 0.2209)	-3.13714
HS105	(0.49877, 0.2039, 103.609, 205.519, 216.046786, 14.76842, 21.043885, 9.998657)	1532.93

A.1 Seeds table

This Appendix presents the seeds used for SOLAR and NOMAD.

Table A.5 Seeds selected for SOLAR.

Seeds' ID	Seeds
#1	1
#2	2
#3	3
#4	4
#5	5

Table A.6 Seeds selected for NOMAD.

Seeds' ID	Seeds
#1	1
#2	2
#3	3
#4	4
#5	5