UNIVERSITÀ
DEGLI STUDI
FIRENZE

Università degli Studi di Firenze
Facoltà di Ingegneria

Corso di Laurea Magistrale in
Ingegneria Informatica

A Branch and Bound Algorithm for the Global Optimization and its Improvements

Relatore:
Prof. Marco Sciandrone
Università degli Studi di Firenze

Autore:
Alberto Guida

Correlatori:
Prof. Fabio Schoen
Università degli Studi di Firenze

Prof. Coralia Cartis
University of Oxford

Anno Accademico 2014-2015

Contents

Acknowledgements

At the end of a long path as can be a university career, people whose feels the need to thank for their support during all these years are a lot.

I would like to express thanks to Mr. Marco Sciandrone and Mr. Fabio Schoen to the great availability and courtesy they have shown, and for the invaluable assistance provided in the preparation of this thesis. A special thanks to Mrs. Coralia Cartis for having worked side-by-side with me, besides the opportunity to live in the beautiful city of Oxford and work at the Mathematical Institute. I am also grateful to Mr. Marco Locatelli for sharing his papers and knowledge about Global Optimization with me.

Finally, some acknowledgements to my family and friends. My heartfelt thanks for my parents Luciana and Ciro, my brother Tommaso, my grandparents Luisa and Tommaso. Further thanks goes to Serena, Daniele, Fabio and Andrea for their constant support and for simply being around.

Alberto

Abstract

This work focuses on Lipschitz optimization with first- and second-order models (particularly cubic regularisation models) using an Overlapping Branch and Bound (oBB) framework based on Fowkes et al. [25]. Particularly, oBB is a Lipschitz derivative based approach that uses an overlapping covering of balls rather than rectangular partitions. The main idea behind oBB is to recursively split an initial ball covering the domain $\mathcal{D}$ into subballs until we find a ball (or balls) of sufficiently small size containing the global minimiser of f over $\mathcal{D}$. Overlapping ball coverings result in an increased number of subproblems that need to be solved and hence requires their parallelization. Cartis et al. [10] develop parallel variants based on both data- and task-parallel paradigms addressing the problem of balancing the load between processors as well. Additionally, novel bounding procedures are implemented. The approach taken in exam, even with parallelization, is not efficient enough to solve large size problems because of the high number of subproblems that need solving due to the special covering (with Euclidean balls) that it is used. With the goal of lessen the computational time of the algorithm allowing the subproblems to be still tractable, a domain reduction method has been designed and implemented. Such strategy, which is considered optional for what concern the convergence of a Branch and Bound algorithm, has allowed to obtain considerable improvements by a computational point of view. Nonetheless, several issues of geometric and structural nature have been addressed. The outline of the paper is as follows. In Chapter 1 we give an introduction on global optimization, describing the different methods used to solve a global optimization problem and the most known applications that fall in this field. We then describe, in Chapter 2, the Branch and Bound methods giving a general framework with all its main phases. In Chapter 3 we consider the particular instance, introduced above, of Branch and Bound strategy for the global optimization of a twice differentiable nonconvex objective function with a Lipschitz continuous Hessian over a compact, convex set. Improvements of this approach based on bounding procedures and parallelization strategies are then described. A general description of domain reduction approaches with all its requirements, is presented in Chapter 4, while improvements obtained with their application and the related issues are finally presented in Chapter 5.

Chapter 1

The term optimization

Optimization is a field of applied mathematics that aim to the goal of finding the extremal value of a function in a domain of definition, subject to various constraints on the variable values. Typically, the problems are modelled in terms of finding the best decision which corresponds to the minimum (or maximum) of a suitable objective function, while it satisfies a given collection of feasibility constraints. The objective function describes overall system performance, such as profit, utility, loss, risk, or error, while the constraints are originated in relationship with physical, technical, economic or some other considerations.
When both the minimum cost configuration and the constraints are linear functions of the decision variables the problem fall in the class of Linear Programming (LP). The well-known Simplex Algorithm (Dantzig [12]) is the most celebrated one for solving these kinds of problems; its complexity in the worst-case is exponential in the number of problem variables, but in most practical instances it performs extremely efficiently. Unlike, one other technique for solving LP problems, the Ellipsoid Algorithm, exhibits polynomial complexity.
Sometimes the structure of the problem explicitly requires integer or binary decision variables. MILP (Mixed-Integer Linear Programming) problems, are problems where some or all variables are integer-valued and the objective function and the constraints are linear. In these problems, because of the integrality constraints on the variables can be expressed as nonlinear functions, the problem itself is nonlinear, despite the linearity of the functions. The techniques for solving MILPs are very different from those for solving LPs, indeed, in most cases, at each step of an algorithm that solves a MILP is required the solution of an entire LP problem. Cutting Plane and Branch and Bound are the two most common algorithms employed in solving MILPs and they have an exponential complexity in the size of the instance in worst-case.
A large number of models naturally belong to the traditional class of the

continuous optimization: in particular, to linear and convex programming. Even so, there exist also numerous cases in which the necessary structural (linearity or general convexity) requirements are not satisfied or not verifiable. On the basis of these structural requirements it is possible evaluate the difficulty of an optimization problem: particularly, the true characteristic discriminant between "easy" and "hard" optimization problems is the convex/nonconvex issue. A problem is convex if it is a minimization of a convex function over a convex set. A well-known result says that, given a convex problem, a locally optimal solution is also globally optimal. However, nonconvex problems may have many different local optimal, and the choice of the best one is an hard task. The field that studies extremal locations of nonconvex functions subject to (possibly) nonconvex constraints is called Global Optimization.

Many hard optimization problems, such as the traveling salesman problem or the protein folding problem, are global optimization problems. Even though, the truth of the unresolved conjecture $P \neq NP$ [26] would imply that there are no general algorithms that solve a given global optimization problem in polynomial time; some global optimization problems have been solved by present methods. In the next section, further details on global optimization will be given, and different resolution methods will be described.

1.1 Global Optimization

The aim of global optimization, as it has been described above, is to find the globally best solution of models, in the presence of multiple local optima. Nonlinear models are located in many applications, e.g., in biotechnology, data analysis, financial planning, process control, risk management and so on. These quite complex nonlinear systems, have an associated decision models that may have multiple locally optimal solutions. In most realistic cases, because of the number of such local solutions is not known a priori, and the quality of global and local solutions may differ substantially, these decision models can be very difficult. Thus, it is not possible directly apply standard optimization strategies to solve them.

To formulate a problem of global optimization, consider the follow non linear problem (NLP):

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

where $f : \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}$ is smooth and in general non-convex and $\mathcal{D}$ is a compact, convex set. Typically, the set $\mathcal{D}$ is a subset of $\mathbb{R}^n$ defined by constraints $g(x)$, where g is a set of m possibly nonlinear functions of x. Moreover, it is often useful to express the variable bounds as $x_L \leq x \leq x_U$ $(x_L, x_U \in \mathbb{R}^n)$. In some cases several variables may be constrained to

only take integer values ($x_i \in \mathbb{Z}$ for all i in an index set $Z \subseteq \{1, ..., n\}$) leading to the class of global optimization Mixed-Integer Nonlinear Programming problem (MINLP)

$$\begin{aligned} \min_x f(x) \\ g(x) \leq b \\ x_L \leq x \leq x_U \\ x_i \in \mathbb{Z} \ \ \forall \ i \in Z. \end{aligned}$$

Given a global optimization problem, we would like to find

$$f^* = \min_{x \in \mathcal{D} \subseteq \mathbb{R}^n} f(x) \tag{1.2}$$

and

$$x^* = \arg \min_{x \in \mathcal{D} \subseteq \mathbb{R}^n} f(x). \tag{1.3}$$

A global minimum or global optimum is any $x^* \in D$ such that

$$f(x^*) \leq f(x) \ \ \forall x \in \mathcal{D}. \tag{1.4}$$

A point $\hat{x}$ is a local optimum if exists a constant $\epsilon > 0$ such that

$$f(\hat{x}) \leq f(x) \ \ \forall x \in \mathcal{D} \cap \mathcal{B}(\hat{x}, \epsilon) \tag{1.5}$$

where $\mathcal{B}(\hat{x}, \epsilon) = \{x \in \mathbb{R}^n : ||x - \hat{x}|| \leq \epsilon\}$ is a ball in $\mathbb{R}^n$. Obviously, any global optimum is also a local optimum, but the opposite is generally false.

If are used traditional local search methods to solve a global optimization problem, then depending on the starting point of the search, it could be frequently found locally optimal solutions. Hence, a global search method is needed to find the globally optimal solution. Several of the most important global optimization strategies are listed below and most software implementations are based on one or more of these strategies.

Exact methods include:

1. Naive (passive) approaches: include passive or direct sequential global optimization strategies such as uniform grid and pure random searches. Such methods are convergent under analytical assumptions but they are are not practical in higher-dimensional problems (Pintér [55], Zhigliavsky and Pintér [73]).

2. Complete (enumerative) search methods: based on an exhaustive enumeration of all possible solutions. They are applicable to combinatorial optimization problems, and to certain structured continuous global optimization problem such as concave programming (Horst and Tuy [31]).

3. Homotopy (parameter continuation) and trajectory strategies: they have the objective of visiting all stationary points of the objective function leading to the list of all optima. This approach is applicable to smooth GO problems and includes differential equation model based, path-following search strategies and more (Diener [15], Forster [24]).

4. Successive approximation (relaxation) methods: in this approach, the problem is replaced, through a successive refinement process, by a sequence of relaxed sub-problems that are easier to solve (e.g., by cutting planes and by more general cuts). These algorithms are applicable to different structured GO such as concave minimization, or differential convex problems (Horst and Tuy [31]).

5. Branch and Bound algorithms: strategies based upon adaptive partition, sampling, and bounding procedures to solve GO models; in particular, these operations are applied iteratively to the collection of active subsets (also called "candidate") within the feasible set $\mathcal{D}$. These general methods typically rely on some a priori structural knowledge about the problem such as how rapidly each function can vary (e.g., the knowledge of a suitable overall Lipschitz constant). This general methodology is applicable to wide classes of global optimization problems, such as concave minimization, reverse convex programs, DC programming, and Lipschitz optimization (Neumaier [50], Hansen 1992, Ratschek and Rokne [58], Kearfott [33], Horst and Tuy [31], Pintér [55]).

6. Bayesian search (partition) algorithms: these methods are based on statistical information, that allow to have a stochastic description of the function-class modelled. Based upon this model and the actual search results, the characteristics of problem-instance are adaptively estimated and updated during optimization. This general approach is applicable to general continuous global optimization problems (Mockus et al. [45]).

7. Adaptive stochastic search algorithms: these procedures are based on random sampling within the feasible set. Nonetheless the basic form includes various random search strategies that are convergent with probability one; enhancements such as adaptive search strategy adjustments, sample clustering, deterministic solution refinement options and statistical stopping rules, can also be added. The methodology is applicable to both discrete and continuous global optimization problems under general conditions (Boender and Romeijn [7], Pintér [55], Zhigliavsky and Pintér [73]).

8. Integral methods: these methods allow to approximate the level sets of the objective function f to reach the determination of the essential

supremum of f over $\mathcal{D}$ (Zheng and Zhuang [72], Hichert et al. [29]).

Heuristic strategies include:

1. "Globalized" extensions of local search methods: the main idea of these practical strategies are based on a preliminary global search (passive grid or random search) phase, followed by applying local scope search (random multistart). Such strategies are applicable to smooth GO problems. Frequently, enhancements are added to this strategy: e.g., the clustering of sample points attempts to select only a single point from each sampled of f from which a local search method can be started. (Pintér [55], Zhigliavsky and Pintér [73]).

2. Genetic algorithms, evolution strategies: these methods heuristically simulate biological evolution. Based on a "population" of candidate solution points, an adaptive search procedure is applied. The poorer solutions are dropped with a competitive selection, while the remaining pool of candidates with higher value are then recombined with other solutions by swapping components with another. Based on diverse evolutionary "game rules", a variety of deterministic and stochastic algorithms can be constructed. These strategies are applicable to both discrete and continuous GO problems under structural requirements (Michalewicz [44], Osman and Kelly [54], Glover and Laguna [28], Voss et al. [70]).

3. Simulated annealing: based on the physical analogy of cooling crystal structures that try to reach some stable equilibrium (globally or locally minimal potential energy); this general principle is applicable to both discrete and continuous global optimization problems under structural requirements (Osman and Kelly [54], Glover and Laguna [28]).

4. Tabu search: in this meta-heuristic technique, the main idea is to start from an initial solution and perform a set of moves that lead to a new solution inside the neighborhood of the current one forbidding of moving in points already visited in the search space, i.e., in order to avoid paths already investigated, one can temporarily accept new inferior solutions. Primarily applied to combinatorial optimization problems, Tabu search can also be extended to handle continuous global optimization problems through his discrete approximation (Osman and Kelly [54], Glover and Laguna [28], Voss et al. [70].

5. Approximate convex global underestimation: based on directed sampling in $\mathcal{D}$, this heuristically strategy aims to estimate convexity characteristics of the objective function f. These approaches are applicable to smooth problems (Dill et al. [16]).

6. Continuation methods: generally these approaches perform in two phases; first they transform the objective function into a more smooth function with easily calculated local minimizers, and then use a local minimization procedure to trace all minimizers back to the original function. Once again, this methodology is applicable to smooth problems (Moré and Wu [48]).

7. Sequential improvement of local optima: in order to find better optima these methods usually build auxiliary functions to assist the searching procedure. The general heuristic principle includes tunneling, deflation, and filled function approaches. These strategies are applicable to smooth global optimization problems (Levy and Gomez [36]).

Based on these different strategies, the different algorithms can be classified according to the goal they should reach:

- An incomplete method uses efficient heuristics for searching but can terminate in a local minimum;
- An asymptotically complete method reaches a global minimum with certainty (at least with probability one) if it runs for a period indefinitely long, but it is not able to know when a global minimizer has been found;
- A complete method reaches a global minimum with certainty, assuming an indefinitely long run time, and knows after a finite time that an approximate global minimizer has been found (given a certain tolerance);
- A rigorous method reaches a global minimum with certainty and within given tolerances even in the presence of errors.

Note that global optimization is "hopeless": without global information no algorithm will find a certifiable global optimum unless it generates a dense sample. A rigorous definition of global information might be provided some examples of global information such as:

- number of local optima;
- global optimum value;
- for global optimization problems over a box, (an upper bound on) the Lipschitz constant

$$|f(y) - f(x)| \leq L||x - y|| \ \ \forall x, y;$$

- concavity of the objective function and convexity of the feasible region;
- an explicit representation of the objective function as the difference between two convex functions (and convexity of the feasible set).

1.2 Why global optimization?

Superficially, global optimization is just a stronger version of local optimization where instead of searching for a locally feasible point, the interest is focused in identifying the globally best point in the feasible region. In many practical applications, finding the globally best point is desirable but not essential, indeed, for such problems, an incomplete search can be performed. However, there are a number of problem classes where it is indispensable to apply a complete search. This is in particular the case for:

- hard feasibility problems, where local strategies do not yield useful information since they typically fall in local minimizers of the objective function, not providing feasible points;
- safety verification problems, where one can have a severe underestimation of the true risk treating non global extrema as worst cases;
- many problems in chemistry, where often only the global minimizer (of the free energy) matches to the situation correspondent in reality;
- semi-infinite programming, where global minimizers of auxiliary problems is usually involved by the optimal configurations.

To show the relevance of global optimization for both pure and applied mathematics, below typical applications are presented:

(i) Many problems in graph theory are global optimization problems. For example, the maximum clique problem requires to find in a given graph the maximal number of mutually adjacent vertices. An equivalent formulation, given by a well-known theorem of Motzkin and Strauss [49], is the indefinite quadratic program

$$\begin{gathered} \max\ x^T A x \\ e^T x = 1;\ x \geq 0; \end{gathered}$$

where A is the adjacency matrix of the graph and e is the all-one vector. Since the maximum clique problem is NP-hard, all classes of global optimization problems that contain indefinite quadratic programming hold the same complexity.

(ii) Packing problems. The problem is to place a number of k-dimensional objects within a number of larger regions of k-space (both object and region of known shape) in such a way that there is no overlap and a measure of the wastage is minimized. Many packing problems rise in industry; but there are also a number of famous packing problems in geometry, of which the most famous problem of finding the densest packing of equal balls in Euclidean d-space. The problem can be formulated as follow

$$\begin{array}{rl} \max & r \\ ||X_i - X_j|| \geq 2 & \forall i \neq j \\ X_i^{(k)} \geq r & \forall i, k \\ X_i^{(k)} \leq 1 - r & \forall i, k \end{array}$$

This problem is equivalent to the maximal dispersion problem:

$$\begin{array}{rl} \max & D \\ ||X_i - X_j|| \geq D & \forall i \neq j \\ X_i \in [0,1]^d & \forall i \end{array}$$

where, $d = 2 \Rightarrow$ circle packing while $d = 3 \Rightarrow$ sphere packing.

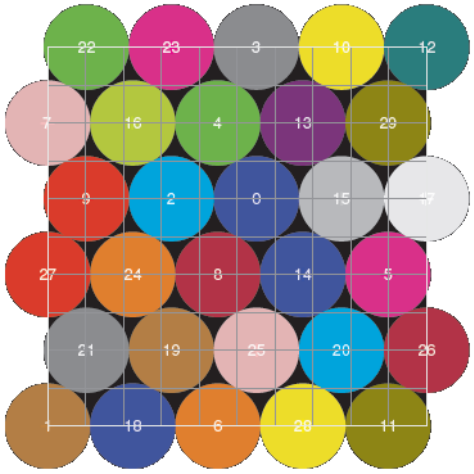

Figure 1.1: Disk and sphere packing problem.

(iii) Scheduling problems. The problem is to match tasks (or people) and slots (time intervals, machines, rooms, airplanes, etc.) such that every slot handles exactly one task. While linear assignment is a simple scheduling problem that can be formulated as linear program and is solved very efficiently, the related quadratic assignment problem is one of the hardest global optimization problems, cf. Anstreicher [3].

(iv) Molecular conformation problem. This problem consist in finding the minimum conformation of complex molecules. Protein folding [51] is a sample of this problem and has the objective of finding the equilibrium configuration of the N atoms in a protein molecule (with a given amino acid sequence), assuming that the forces between the atoms are known. These forces are given by the gradient of the $3N$-dimensional potential energy function $V(y_1, ..., y_N)$, where y_i denotes the coordinate vector of the ith atom,

while the equilibrium configuration is given by the global minimization of V. Because of the presence of short-range repulsive forces, numerous local minima are present.

(v) Chemical equilibrium problems (Floudas [21], McDonald and Floudas [42]), and application in robotic (Neumaier [52]).

1.3 The structure of Global Optimization Algorithms

In most global optimization algorithms it is possible to identify two distinct phases: a global phase that it is delegated to exhaustive exploration of the search space and a local phase called at each iteration of the global phase to identify a locally optimal point. Note that, the global phase calls the local procedure as a "black-box": therefore, for a fast convergence a reliable and robust local procedure is needed.
Since local optimization of NLPs is an NP-hard problem in itself, so finding the global optimum of most nonconvex problems is also NP-hard. Although the local optimization phase is a blackbox function call, a good knowledge of the local optimization technique used is important to understand the global phase, so in next section some information to the theory of local optimization of NLPs will be given.

1.4 Local Optimization of NLPs

For giving an overview of local optimization, first it is important introduce some basic notions of convex analysis, explaining the necessary and sufficient conditions for local optimality. After that, an iterative approach to find a local minimum of a given NLP is described.

1.4.1 Notions of convex analysis

Definition 1. A set $\mathcal{S} \subseteq \mathbb{R}^n$ is convex if for any two points $x, y \in \mathcal{S}$ the segment between them is wholly contained in $\mathcal{S}$, that is, $\forall \lambda \in [0,1]\, \lambda x + (1-\lambda)y \in \mathcal{S}$.

A linear equation $a^T x = b$ where $a \in \mathbb{R}^n$ defines a hyperplane in $\mathbb{R}^n$. The corresponding linear inequality $a^T x \leq b$ defines a closed half-space. Both hyperplanes and closed half-spaces are convex sets. Since any intersection of convex sets is a convex set, the subset of $\mathbb{R}^n$ defined by the system of closed half-spaces $Ax \geq b, x \geq 0$ is convex (where $A \in \mathbb{R}^{m \times n}, b \in \mathbb{R}^m$).

Definition 2. A subset $\mathcal{S} \subseteq \mathbb{R}^n$ defined by a finite number of closed halfspaces is called a polyhedron. A bounded, non-empty polyhedron is a polytope.

Having defined convex sets, it is now possible turn our attention to convex functions.

Definition 3. A function $f : X \subseteq \mathbb{R}^n \to \mathbb{R}$ is convex over a convex set X if for all $x, y \in X$ and for all $\lambda \in [0, 1]$ we have:

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).$$

The main theorem in convex analysis, says that a local minimum of a convex function over a convex set is also a global minimum.

Theorem 1. *Let $X \subseteq \mathbb{R}^n$ be a convex set and $f : X \to \mathbb{R}$ be a convex function. Given a point $x^* \in X$, suppose that there is a ball $\mathcal{B}(x^*, \epsilon) \subset X$ such that for all $x \in \mathcal{B}(x^*, \epsilon)$ we have $f(x^*) \leq f(x)$. Then $f(x^*) \leq f(x)$ for all $x \in X$.*

1.4.2 Necessary and sufficient conditions for local optimality

In this section we present necessary and sufficient conditions for a feasible point x^* to be a locally optimal point. Let us consider the optimization problem as in Eq. (1.1) where the constraint's relations R are all equalities, the variables are unbounded ($x_L = -\infty, x_U = \infty$) and the index set Z is empty, i.e., consider the following nonlinear (possibly nonconvex programming) problem with continuous variables defined over $\mathbb{R}^n$:

$$\begin{aligned} &\min_{x \in \mathbb{R}^n} f(x) \\ &g(x) = 0, \end{aligned} \tag{1.6}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are C^1 (i.e., continuously differentiable) functions.
A *constrained critical point* of problem (1.6) is a point $x^* \in \mathbb{R}^n$ such that $g(x^*) = 0$ and the directional derivative of f along g is 0 at x^*.

Given a scalar function $f : \mathbb{R}^n \to \mathbb{R}$ we can define the function vector ∇f as $(\nabla f)(x) = \left(\frac{\partial f(x)}{\partial x_1}, ..., \frac{\partial f(x)}{\partial x_n}\right)^T$, where $x = (x_1, ..., x_n)^T$. If g is a vector-valued function $g(x) = (g_1(x), ..., g_m(x))$, then ∇g is the set of function vectors $\{\nabla g_1, ..., \nabla g_m\}$. From these notions we can show the following result:

Theorem 2 (Lagrange Multiplier Method)**.** *If x^* is a constrained critical point of problem (1.6), $m \leq n$, and $\nabla g(x^*)$ is a linearly independent set of vectors, then $\nabla f(x^*)$ is a linear combination of the set of vectors $\nabla g(x^*)$.*

By Theorem (2) there exist scalars $\lambda_1, ..., \lambda_m$, such that

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) = 0.$$

The above condition is equivalent to saying that x^* is a critical point of the function f if x^* is a constrained critical point of f s.t. $g(x^*) = 0$:

$$L(x, \lambda) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x). \tag{1.7}$$

Function (1.7) is called the Lagrangian of f w.r.t. g, and $\lambda_1, ..., \lambda_m$ are called the Lagrange multipliers.

To explain the necessary conditions for local minimality, first it is necessary consider the well known theorem of alternatives called Farkas' Lemma. The following three results are necessary to introduce the Lemma, which will be defined later.

Theorem 3 (Weierstrass)**.** *Let $\mathcal{S} \subseteq \mathbb{R}^n$ be a non-empty, compact set and let $f : \mathcal{S} \to \mathbb{R}$ be continuous on $\mathcal{S}$. Then there exist a global minimum point of f in $\mathcal{S}$.*

Proposition 1. *Given a non-empty, closed convex set $\mathcal{S} \subseteq \mathbb{R}^n$ and a point $x^* \notin \mathcal{S}$, there exists a unique point $x' \in \mathcal{S}$ with minimum distance from x^*. Furthermore, x' is the minimizing point if and only if $\forall x \in \mathcal{S}$ we have $(x^* - x')^T(x - x') \leq 0$.*

Proposition 2. *Given a non-empty, closed convex set $\mathcal{S} \subseteq \mathbb{R}^n$ and a point $x^* \notin \mathcal{S}$, there exists a separating hyperplane $h^T x = d$ (with $h \geq 0$) such that $h^T x \leq d \ \forall x \in \mathcal{S}$ and $h^T x^* > d$.*

Theorem 4 (Farkas' Lemma)**.** *Let A be an $m \times n$ matrix and c be a vector in $\mathbb{R}^n$. Then (exactly) one of the following systems has a solution:*

1. *$Ax \leq 0$ and $c^T x > 0$ for some $x \in \mathbb{R}^n$;*
2. *$\mu^T A = c^T$ and $\mu \geq 0$ for some $\mu \in \mathbb{R}^m$.*

Proof. Suppose system (2) has a solution, then $\mu^T Ax = c^T x$. Supposing $Ax \leq 0$, since $\mu \geq 0$, we have $c^T x \leq 0$. Conversely, suppose system (2) has no solution. Let $Im_+(A) = \{z \in \mathbb{R}^n : z^T = \mu^T A, \mu \geq 0\}$ be a convex set and $c \notin Im_+(A)$. By Prop. (2), there is a separating hyperplane $h^T x = d$ such that $h^T z \leq d \ \forall z \in Im_+(A)$ and $h^T c > d$. Since $0 \in Im_+(A)$ and $d \geq 0$ we have $h^T c > 0$. Furthermore, $d \geq z^T h = \mu^T Ah \ \forall \mu \geq 0$. Since μ can be arbitrarily large, $\mu^T Ah \leq d \Rightarrow Ah \leq 0$. Taking $x = h$ we have that x solves system (1). □

We can now consider the necessary conditions for local minimality. Consider the following problem:

$$\begin{aligned} &\min_{x\in\mathbb{R}^n} f(x) \\ &g(x) \leq 0, \end{aligned} \tag{1.8}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are C^1 functions. A constrained minimum of problem (1.8) is a minimum x^* of $f(x)$ such that $g(x^*) \leq 0$. It can be shown that if x^* is a constrained minimum then there is no non-zero feasible descent direction at x^*.

Definition 4. Let $x \in \mathcal{S}$ an assigned point. A vector $d \in \mathbb{R}^n,\ d \neq 0$ is a feasible direction in x if exists $\bar{t} > 0$ such that

$$x + td \in S \ \ \forall t \in [0, \bar{t}]$$

Definition 5. Let $x \in \mathcal{S}$ an assigned point. A vector $d \in \mathbb{R}^n,\ d \neq 0$ is a descent direction for f in x if exists $\bar{t} > 0$ such that

$$f(x + td) < f(x) \ \ \forall t \in (0, \bar{t}]$$

Under hypothesis of continuity of gradient we have a characterization of descent directions

Proposition 3. *Assuming the gradient is continue on $\mathbb{R}^n$ and let $x \in \mathbb{R}^n$. If*

$$\nabla f(x)^T d < 0 \tag{1.9}$$

then d is a descend direction for f in x.

Remark 1. If $\nabla f(x)^T d > 0$ the direction d is a climb direction, while if $\nabla f(x)^T d = 0$ none conclusion ca be do. Furthermore, note that the previous condition is even necessary if f is a convex function, since in this case $\forall i \geq 0$ we have

$$f(x + td) \geq f(x) + t\nabla f(x)^T d.$$

Theorem 5 (Karush-Kuhn-Tucker Conditions)**.** *If x^* is a constrained minimum of problem (1.8), I is the maximal subset of $\{1, ..., m\}$ such that $g_i(x^*) = 0\ \forall i \in I$, and $\nabla\bar{g}$ is a linearly independent set of vectors (where $\bar{g} = \{g_i(x^*) : i \in I\}$), then $-\nabla f(x^*)$ is a conic combination of the vectors in $\nabla\bar{g}$, i.e. there exist scalars $\lambda_i\ \forall i \in I$ such that the following conditions hold:*

$$\nabla f(x^*) + \sum_{i\in I} \lambda_i \nabla g_i(x^*) = 0 \tag{1.10}$$

$$\forall i \in I\ (\lambda i \geq 0). \tag{1.11}$$

Proof. Since x^* is a constrained minimum and $\nabla\bar{g}$ is linearly independent, there is no nonzero feasible descent direction at x^* such that $(\nabla\bar{g}(x^*))^T d \leq 0$ and $-\nabla f(x^*)d > 0$. By a direct application of Farkas'Lemma, there is a vector $\lambda \in R^{|I|}$ such that $\nabla(\bar{g}(x^*))\lambda = -\nabla f(x^*)$ and $\lambda \geq 0$. □

The KKT conditions (1.10) and (1.11) can be reformulated with the following:

$$\nabla f(x^*) + \sum_{i=1}^{m} \lambda_i \nabla g_i(x^*) = 0 \tag{1.12}$$

$$\forall i \leq m \ \ (\lambda_i g_i(x^*) = 0) \tag{1.13}$$

$$\forall i \leq m(\lambda_i \geq 0). \tag{1.14}$$

By defining $\lambda_i = 0 \ \forall i \notin I$ this riformulation is easily verifiable. Conditions (1.13) express the fact that if a constraint is not active at x^* then its corresponding Lagrange multiplier is 0 and they are called *complementary slackness conditions*and they. Furthermore, we can say that a point x^* satisfying the KKT conditions is a KKT point, or KKT solution.

Consider now the same general NLP (1.8) with the addition of the equality constraints:

$$\begin{aligned} \min_{x\in\mathbb{R}^n} f(x) \\ g(x) \leq 0 \\ h(x) = 0, \end{aligned} \tag{1.15}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^m$ and $h : \mathbb{R}^n \to \mathbb{R}^p$ are C^1 functions. A constrained minimum of problem (1.15) is a minimum x^* of $f(x)$ such that $g(x^*) \leq 0$ and $h(x) = 0$.
By applying theorems (2) and (5), we can define the Lagrangian of problem (1.15) by the following:

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i g_i(x) + \sum_{i=1}^{p} \mu_i h_i(x), \tag{1.16}$$

and the corresponding KKT conditions as:

$$\begin{aligned} \nabla_x L(x, \lambda, \mu) = 0 \\ \lambda_i g_i(x^*) = 0 \ \ \forall i \leq m \\ \lambda_i \geq 0 \ \ \forall i \leq m. \end{aligned} \tag{1.17}$$

In most practice cases for local optimality are frequently used the sufficient conditions witch say that if a point x^* is a KKT point and f is convex in a neighbourhood of x^*, then x^* is a local minimum.

1.5 A brief history of global optimization

In this section we present the evolution that global optimization has had during the history. In particular, because of the goal of this paper, major focus is given to Branch and Bound (B&B) strategies.
Generic optimization problems have been important throughout history in engineering applications and so on. The first significant work in optimization was achieved by Lagrange in 1797. Nevertheless, before the introduction of electronic computers, global optimality of solutions was guaranteed only for the local optimization of convex functions, a rather limited class of problems. Markowitz and Manne [40] in 1957, and Dantzig et al. [13, 14] in 1958 and 1960, used piecewise linear approximations for the approximate global minimization of separable nonconvex programs, formulating them as mixed integer linear programs. The methods that, with the advent of first electronic computers, were first used in global optimization were deterministic techniques, mostly based on the divide-and-conquer principle. Instead of trying to locate a minimum by solving a set of equations by symbolic/algebraic methods, one would try to build a sequence of approximate solutions which converges to the original solution by dividing the problem into smaller subproblems. B&B is one typical algorithm which embodies the divide-and-conquer principle. Several B&B techniques for discrete optimization, applicable to mixed integer linear programs, were introduced by Land and Doig [35] in 1960. Because of the nature of the algorithm the B&B algorithm applies very well to cases where variables are discrete in nature and thus, naturally decomposable. In fact, the first applications of B&B to global optimization problems were related to discrete problems such as the Travelling Salesman Problem in 1963 (see Little et al. [38]). Motzkin and Strauss [49] showed in 1965 that solving the (discrete) maximum clique problem is equivalent to finding the global minimum (or maximum) of a special nonconvex quadratic program. In 1969, Falk and Soland [20] gave the first piecewise linear relaxations of nonconvex problems, useful for obtaining bounds in a B&B scheme. In 1972, McCormick [41] introduced the now frequently used linear relaxations for products and quotients, which made accessible to the B&B technique, the solution of general factorable global optimization problems. Moore [46], following an unpublished technical report by Moore and Yang [47] in 1959, showed in 1962 in Part 4 of his Ph.D. thesis that by repeating the subdivision and by making an interval evaluation, the range, and hence in particular the global minimum of a rational function over a box can be determined in principle with arbitrary accuracy. Skelboe [63] improved, in 1974, this basic but excessively slow method by embedding it into a B&B scheme for continuous variables, giving (what is now called) the Moore-Skelboe algorithm. Moore's thesis [46] also showed that interval methods can be used to prove the non-existence of solutions of nonlinear systems in a box (which nowadays is used to discard boxes in

a B&B scheme) and to reduce the region where a solution can possibly lie (which is now used to avoid excessive splitting). Furthermore, Kahan [32] discovered in 1968 that interval techniques can also be used to prove the existence of solutions of nonlinear systems in a box (and hence to verify feasibility), while Piyavskii [56] in 1972 introduced complete global optimization methods based on Lipschitz constants. Becker and Lago [5] first used clustering methods in 1970, while Torn [68] suggested in his 1974 thesis to combine these with local optimization, defining the most efficient class of stochastic global optimization algorithms. Holland [30] introduced in 1973 genetic algorithms, a popular stochastic heuristics for global optimization. Since the beginning of the 1990s, an explosion of papers, books, algorithms and resources about deterministic and stochastic global optimization, it has been witnessed from the research community. It is significant that one of the first and most widely used book in global optimization [31] does not even mention nonconvex NLPs. Between 1995 and 1996 Ryoo and Sahinidis proposed a Branch and Reduce algorithm [59, 60], the first method that was able to deal directly with the generic nonconvex NLPs. Shortly afterwards, Floudas' team published their first article on the α-BB (Branch and Bound) method [23]. One considerable limitation of the α-BB algorithm is that it relies on the functions being twice differentiable in the continuous variables. After that, a number of Branch and Select algorithms oriented toward the most generic nonconvex MINLP formulation appeared in the literature, such as Smith and Pantelides' symbolic reformulation approach [64, 65, 66], Pistikopoulos' Reduced Space B&B approach [17] and Grossmann's Branch and Contract algorithm [71] (which they only apply to continuous NLPs), Barton's Branch and Cut framework [4] and modern Interval Analysis based on global optimization methods [69, 53, 74] which can be included into the Branch and Select strategy.

Chapter 2

Branch and Bound methods

As it has been introduced in the previous chapter, we are interested in the application of Branch and Bound (B&B) methods. In this chapter we discuss these approaches for solving a GO problem. Under suitable assumption, such approaches allow us to detect in a finite time or, at least, to converge to a globally optimal solution of the problem. Particularly we will consider problems with the form

$$\begin{aligned} P(\mathcal{D}) : \min f(x) \\ g_i(x) \leq 0 \quad i = 1, .., m \\ x \in \mathcal{D} \end{aligned} \tag{2.1}$$

with $f : \mathcal{D} \subset \mathbb{R}^n \to \mathbb{R}$ and all functions $f, g_i, i \in \mathcal{I} = \{1...., m\}$ are assumed to be at least continuously differentiable on a compact, convex set $\mathcal{D}$. Let $\mathcal{S}$ be the feasible region of $P(\mathcal{D})$ denoted by

$$\mathcal{S} = \{x \in \mathcal{D} : g_i(x) \leq 0, i \in \mathcal{I}\},$$

the Weierstrass theorem (3) guarantees the existance of a global solution if $S \neq \emptyset$, since it is a compact set. Denoting by f^* the minimum value (with $f^* = +\infty$ if $\mathcal{S} = \emptyset$) the main idea of a B&B scheme is to fix some optimality and feasibility limits of witch we may be satisfied, i.e, we look for a δ–feasible and (ϵ, δ)–optimal solution of wich we give a definition below.

Definition 6. Given a vector $\delta \in \mathbb{R}^m_+$, a point $x \in \mathcal{D}$ is a δ-feasible point for (2.1) if

$$g_i(x) \leq \delta_i, i = 1, ...m.$$

The set of δ-feasible solution is denoted by $\mathcal{S}_\delta$.

A point x^* is called a δ-optimal solution if $x^* \in \mathcal{S}_\delta$ and it such that

$$f(x^*) \leq f(x) \; \forall x \in \mathcal{S}_\delta.$$

Note that, based on the nature of constraints we can have $\delta_i = 0$ (for simple constraints like convex or even linear), or a strictly positive δ_i (for nonconvex constraints).

Definition 7. Given a vector $\delta \in \mathbb{R}^m_+$ and a scalar $\epsilon \geq 0$, a point $x^* \in \mathcal{D}$ is a (ϵ, δ)-optimal solution for (2.1) if it is δ-feasible and

$$f(x^*) \leq f(x) + \epsilon.$$

In particular, if $x^* \in \mathcal{S}$, x^* is called an ϵ-optimal solution and $f^* < \infty$ is obviously true.

B&B methods are applied in order to solve (2.1) as we describe in terms of a general framework in the next section.

2.1 A general B&B Framework

The main idea of B&B methods is heavily based upon the construction of lower and upper bounds. This can be achieved in many ways, for example by the α-BB method, using Lipschitz constants or by exploiting duality. The direct application of interval arithmetic is also possible. See Locatelli and Schoen [39] for more information about these techniques.

Independently of what bounding procedures are chosen, a general version of a B&B framework with its main phases can be achieved as follow.

1. Initialization

Data: $\delta \geq 0, \epsilon \geq 0$
set $\mathcal{C}_0 = \{\mathcal{S}\}, t = 0$
Compute a lower bound $lb(\mathcal{S})$
if $\hat{x} \in \mathcal{S}_\delta$ is available **then**
    set $ub_\delta = f(\hat{x})$
    set $z = \hat{x}$
else
    set $ub_\delta = \infty$
    leave z undetermined
end
if $ub_\delta \leq lb(\mathcal{S}) + \epsilon$ **then**
    if $ub_\delta < \infty$ **then**
        return the (ϵ, δ)-optimal solution z
    else
        return $\mathcal{S} = \emptyset$
    end
else
    Go to Step 2.
end

2. Node Selection

Select a subset $\mathcal{S}_k \in \mathcal{C}_t$ according to some rule.

3. Branching

Subdivide $\mathcal{S}_k$ into $r \geq 2$ subsets $\mathcal{S}_{1k}, ..., \mathcal{S}_{rk}$ such that $\bigcup_{i=1}^{r} \mathcal{S}_{ik} = \mathcal{S}_k$ and set $\mathcal{C}_{t+1} = \mathcal{C}_t \cup \{\mathcal{S}_{1k}, ..., \mathcal{S}_{rk}\} \setminus \{\mathcal{S}_k\}$.

4. Domain Reduction

Possibly reduce the subsets $\mathcal{S}_{1k}, ..., \mathcal{S}_{rk}$ through the application of domain reduction strategies.

5. Lower Bound Computation

Compute a lower bound $lb(\mathcal{S}_{ik}), i = 1, ..., r$, for all the newly generated subsets.

6. Upper Bound Update

let $\mathcal{T}_\delta$ be a (possibly empty) set of δ-feasible points detected during the lower bound computations

if $\min_{y \in \mathcal{T}_\delta} f(y) < ub_\delta$ **then**

$ub_\delta = \min_{y \in \mathcal{T}_\delta} f(y)$

$z \in \operatorname{argmin}_{y \in \mathcal{T}_\delta} f(y)$

end

7. Fathoming

Update $\mathcal{C}_{t+1}$ by discarding all subsets whose lower bound is larger than $ub_\delta - \epsilon$, i.e.,

$$\mathcal{C}_{t+1} = \mathcal{C}_{t+1} \setminus \{\mathcal{Q} \in \mathcal{C}_{t+1} : lb(\mathcal{Q}) \geq ub_\delta - \epsilon\}. \tag{2.2}$$

If available, apply other fathoming rules in order to (possibly) discard further subsets from $\mathcal{C}_{t+1}$.

8. Stopping Rule

if $\mathcal{C}_{t+1} = \emptyset$ **then**

if $ub_\delta < \infty$ **then**

return the (ϵ, δ)-optimal solution z

else

return $\mathcal{S} = \emptyset$

end

else

Set $t = t + 1$ and go back to Step 2

end

Given any B&B algorithm, his evolution can be represented through a so-called BB tree, whose nodes correspond to each subset generated during his execution. Particularly, given a father node $\mathcal{S}_k$ the branching operation at Step 3 generates r child nodes $\mathcal{S}_{1k}, ..., \mathcal{S}_{rk}$. At any iteration t, the collection $\mathcal{C}_t$ will contain all the leaves of the BB tree not yet fathomed while the root node represent the feasible region $\mathcal{S}$. In the following sections the steps of the B&B approach are discussed and the conditions under which the approach is able to detect (or at least to converge) to globally optimal solution are analyzed.

2.2 Initialization step

This step is a quite standard one. The initial collection $\mathcal{C}_0$ (subset of $\mathcal{S}$) is initialized with the set $\mathcal{S}$. Because of the difficulty of identifying feasible point even in the case of a nonempty feasible region some care is needed: therefore, the upper bound value is not the best-observed value of the objective function at feasible points but rather at δ-feasible points. As long as no δ-feasible point is observed, the value of ub_δ is set equal to ∞. The detection of a feasible or δ-feasible point $\hat{x}$ can be performed either during the computation of the lower bound $lb(\mathcal{S})$ or through any available heuristic for the problem.

2.3 Node selection

For the selection of node/subset in $\mathcal{C}_t$ we can use different kinds of rules. One rule is

$$\text{select } \mathcal{S}_k \in \arg\min_{\mathcal{Q}\in\mathcal{C}_t} lb(\mathcal{Q}),$$

i.e., a node/subset with the minimum lower bound value among all those not yet fathomed is chosen. Since the nodes with a small lower bound are more likely to contain good feasible (or δ-feasible) solutions whose detection allows a quick decrease of the upper bound ub_δ and, thus, a quick fathoming of nodes/subsets, it can result helpful explore these nodes. An alternative of this rule that might be called a *best-first* rule, is the *depth-first* rule where the investigation of the BB tree allow to select the left-most leaf among those still in $\mathcal{C}_t$. Obviously, the two mentioned strategies are not the only possible ones. Indeed, we can observe that the *best-* and *depth-first* rules are someway complementary, and a hybrid rule is usually a good option. In the first levels of the BB tree a *best-first* strategy is employed, while in the lower levels a *depth-firs* strategy is employed most of time in order to avoid the explosion of the memory requirements.

2.4 Lower bound update

Given a subset $\mathcal{S}_k$ of the feasible region $\mathcal{S}$, we denote by $lb(\mathcal{S}_k)$ some value satisfying

$$lb(\mathcal{S}_k) \leq \min_{x\in\mathcal{S}_k} f(x). \tag{2.3}$$

When dealing with the nonconvex problem (2.1), one might directly work on its original formulation and in this case the set $\mathcal{S}_k$ is defined as

$$\mathcal{S}_k = \{x \in \mathcal{D} : g_i(x) \leq 0, i = 1, ...m, h_j(x) \leq 0, j = 1, ..., t\},$$

where the constraints $h_j(x) \leq 0$, $j = 1, ..., t$, are (usually) simple constraints added to the original ones, defining $\mathcal{S}$ along the branches which lead from

the root node $\mathcal{S}$ to the current node $\mathcal{S}_k$. After that, the lower bound $lb(\mathcal{S}_k)$ is computed through the solution of a simpler problem where the function $f, g_i, i = 1, ..., m$, are substituted by convex underestimators $\hat{f}^{\mathcal{X}_k}, \hat{g_i}^{\mathcal{X}_k}$, which are valid over some region $\mathcal{X}_k \supseteq \mathcal{S}_k$, that is,

$$\begin{aligned} lb(\mathcal{S}_k) = \min_{x \in \mathcal{D}} \hat{f}^{\mathcal{X}_k}(x) \\ \hat{g_i}^{\mathcal{X}_k}(x) \leq 0, \;\; i = 1, ..., m, \\ h_j(x) \leq 0 \;\; j = 1, ..., t. \end{aligned} \tag{2.4}$$

Of course, the substitution with the underestimator is not necessary for those function that are already convex. Since

$$\hat{\mathcal{S}}_k = \{x \in \mathcal{D} : \hat{g_i}^{\mathcal{X}_k}(x) \leq 0, \;\; i = 1, ..., m, h_j(x) \leq 0 \;\; j = 1, ..., t\} \supseteq \mathcal{S}_k,$$

and

$$\hat{f}^{\mathcal{X}_k} \leq f(x) \; \forall x \in \mathcal{S}_k,$$

the relation (2.3) holds, therefore, the problem (2.4) is a relaxation of the problem $\min_{x \in \mathcal{S}_k} f(x)$. In the case that $\hat{\mathcal{S}}_k = \emptyset$, the lower bound $lb(\mathcal{S}_k)$ is set equal to ∞, so that the node/subset is certainly fathomed.

If one wants to work directly with the original formulation, it is possible to use an alternative way where first the problem (2.1) must be reformulated into some equivalent problem, and then the relaxations of the reformulated problem are considered.

2.5 Upper bound update

The step of updating the upper bound value should take into account the difficulties related even to the detection of feasible solution for problem (2.1). In fact, the computation of the lower bound might deliver some feasible, or at least, δ-feasible points. Given the newly detected δ-feasible points, if at least one of these has a function value lower than ub_δ, then the best function value observed among the newly δ-feasible points is used to update the value ub_δ and the point z defined into the Step 5 is update accordingly. Noting that, unless $\delta = 0$, since δ-feasible points outside $\mathcal{S}$ might have the value of the objective function lower than the optimal value f^*, the value ub_δ is not necessary an upper bound for f^*. Furthermore, we can also make the following remark.

Remark 2. Since points $y \in \mathcal{T}_\delta$ are usually optimal solution of the relaxations solved to compute $lb(\mathcal{S}_{ik})$, we have that $f(y) \geq lb(\mathcal{S}_{ik})$.

In order to possibly update the upper bound we can note that heuristic method can also be applied at nodes of the BB tree.

2.6 Fathoming rule

The standard fathoming rule (2.2) allows to discard a node/subset $\mathcal{S}_k$ if $lb(\mathcal{S}_k) \geq ub_\delta - \epsilon$ because it is guaranteed that no δ-feasible solution with function value lower than $ub_\delta - \epsilon$ can be detected. A graphical representation is showed in Figure

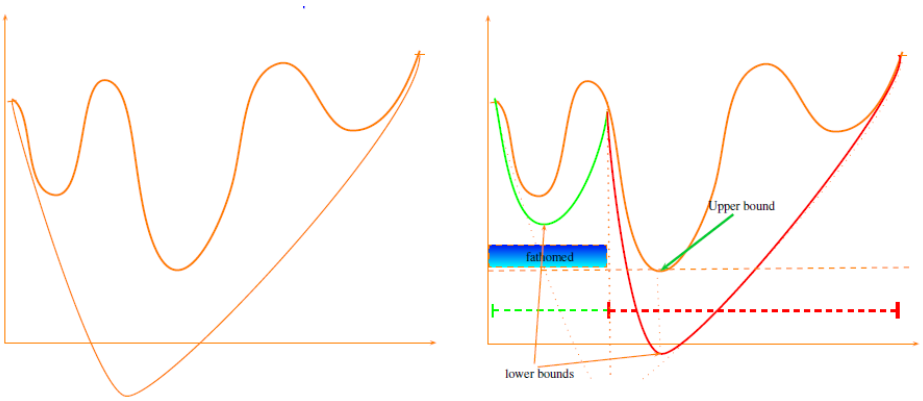

Figure 2.1: Fathoming via upper bound computation.

2.7 Stopping rule

The rule for stopping is the standard one for B&B methods: particularly, we stop all the nodes/subsets of the BB tree that have been fathomed. If $ub_\delta = \infty$ at stopping, this means that the feasible region of the problem is empty (a clear conclusion can be do). When instead $ub_\delta < \infty$, the algorithm returns a δ-feasible solution z for which we can guarantee that, if $\mathcal{S} \neq \emptyset$, z is even a (ϵ, δ)-optimal solution. This uncertainly in the returned value is due to the possible difficulty of detecting feasible points for problems with nonconvex constraints and is equivalent to the difficulty of finding the exact global minimum value for a nonconvex problem. One can also keep track, even if this part is not covered by the pseudo code, of the best feasible solution observed during the execution of the algorithm but,

- there is no guarantee that a feasible solution will be observed even if $\mathcal{S} \neq \emptyset$;
- even if a feasible points are observed, the best objective function value might be far away from the optimal value.

2.8 Branching

The operation of branching allows us to subdivide a given set $\mathcal{S}_k$ into a finite number $r \geq 2$ of subsets $\mathcal{S}_{1k}, ..., \mathcal{S}_{rk}$. If the original set $\mathcal{S}_k$ is defined

by some constraints, each of the subsets $\mathcal{S}_{ik}$ is obtained by adding one or more constraints to those defining $\mathcal{S}_k$, in such way that each point in $\mathcal{S}_k$ belongs to at least one subset $\mathcal{S}_{ik}$, i.e., $\cup_{i=1}^{r} \mathcal{S}_{ik} = \mathcal{S}_k$. Unlike the B&B approaches for integer programs where it is usually guaranteed that each point belongs to exactly one subset, in the field of nonconvex problems a *geometric branching* are often employed. Into this kind of branching, the feasible region of the problem is enclosed within a set $\mathcal{X}_0 \supset \mathcal{S}$ with a given simple geometrical shape, and each subset (corresponding to a node of the BB tree) is enclosed into a set with the same geometrical shape. In other words, this operation subdivides a set with a fixed geometrical shape into subsets with the same shape (Figure 2.2 shows an example). Obviously, the complexity of computing bounds over regions with a certain shape is related to the shape itself.

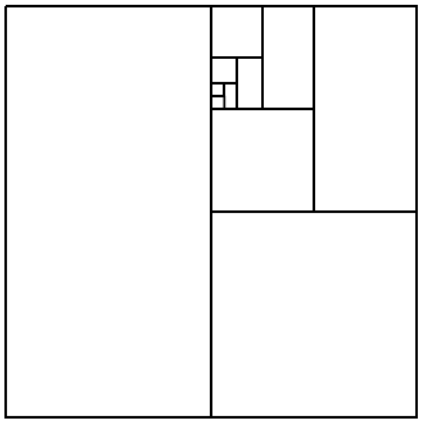

Figure 2.2: Example of nonuniform branching procedure by using boxes.

2.9 Domain reduction

Domain reduction techniques are considered facultative because they are not necessary to guarantee convergence or finiteness results for B&B algorithm. Nevertheless, these techniques are very important in practice and they might have a strong impact on the practical performance. In particular, these techniques are used for cutting the feasible region without cutting the global optimum solution and for this scope, they add simple inequalities to the original problem (also called *cutting planes*), or strengthen existing ones. The added inequalities or the strengthened ones can be classified into two categories

- **Feasibility based**: when they are satisfied by all feasible points. They are usually employed for problems with nonconvex feasible regions in order to get improved convex relaxations of such regions.

- **Optimality based**: when they are satisfied by at least one optimal solution. Moreover, some feasible solutions (and, in same case, also some optimal ones) could not satisfy such inequalities.

An important role in the field of domain reduction is played by *range reduction* techniques, but more general domain reductions strategies con be considered.

Chapter 3

Overlapping Branch and Bound (oBB)

In this chapter we present a B&B Algorithm for the Global Optimization of Hessian Lipschitz Continuous Functions. The focus is aimed in solving the following global optimization problem

$$\min_{x \in \mathcal{D}} f(x) \tag{3.1}$$

where $\mathcal{D} \subset \mathbb{R}^n$ is a compact, convex set and $f : \mathcal{C} \to \mathbb{R}$ is a twice-continuously differentiable nonconvex function defined on a suitable compact set $C \subset \mathbb{R}^n$ containing $\mathcal{D}$. By assuming that f has a Lipschitz continuous Hessian on $\mathcal{C}$, the algorithm is based on applying cubic regularisation techniques to the objective function within an overlapping branch and bound algorithm (oBB) (based on Fowkes et al. [25]), and unlike other B&B algorithms, lower bounds are obtained via nonconvex underestimators of the function.

3.1 Description of the Algorithm

The developed algorithm, described as an extension of the canonical B&B algorithm, defines a branching phase which includes a systematic covering and refinement of $\mathcal{D}$ by balls $\mathcal{B}$, while the bounding procedure requires the computation of both lower and upper bounds on the minimum of f over each $\mathcal{B}$ (see Appendix A for the related notation). To construct a lower bound for the minimum of f over $\mathcal{B}$, global information in the form of a Lipschitz constant is often used. The most simple case is when the lower bound is based on a Lipschitz constant of the objective function f, and thus has the immediate form

$$f(x) \geq f(x_{\mathcal{B}}) - L_f(\mathcal{B})\|x - x_{\mathcal{B}}\|.$$

for some point $x_{\mathcal{B}}$ in a subregion $\mathcal{B}$. A more accurate lower bound based on the gradient Lipschitz constant can be derived using Taylor's theorem to

first order

$$f(x) \geq q_{\mathcal{B}}(x) := f(\mathcal{B}) + (x - x_{\mathcal{B}})^T g(x_{\mathcal{B}}) - \frac{L_g(\mathcal{B})}{2}\|x - x_{\mathcal{B}}\|_2^2. \tag{3.2}$$

for some point $x_{\mathcal{B}}$ in a subregion $\mathcal{B}$. It is also possible to use the second order Taylor's theorem to obtain a cubic lower bound based on the Hessian Lipschitz constant (see Fowkes et al. [25])

$$f(x) \geq c_{\mathcal{B}}(x) := f(x_{\mathcal{B}}) + (x - x_{\mathcal{B}})^T g(x_{\mathcal{B}}) + \frac{1}{2}(x - x_{\mathcal{B}})^T H(x_{\mathcal{B}})(x - x_{\mathcal{B}}) - \frac{L_H(\mathcal{B})}{6}\|x - x_{\mathcal{B}}\|_2^3 \tag{3.3}$$

for some point $x_{\mathcal{B}}$ in a subregion $\mathcal{B}$. To find, instead, an upper bound for the minimum of f over $\mathcal{B}$, we simply evaluate f at some feasible point in $\mathcal{B}$. The main idea behind the algorithm is to recursively split an initial ball covering the domain $\mathcal{D}$ into subballs until we find a ball (or balls) of sufficiently small size containing the global minimiser of $f(x)$ over $\mathcal{D}$. Since we are able to obtain bounds on the minimum of $f(x)$ over any ball in $\mathcal{D}$, we can use them to discard balls which cannot contain the global minimiser, i.e. balls whose lower bound is greater than the smallest upper bound. The complete B&B algorithm is showed below:

Algorithm 1 : Branch and Bound Algorithm for Hessian Lipschitz Optimization

0. Initialization:
   (a) Set $k = 0$.
   (b) Let $\mathcal{B}_0$ be a ball with centre $x_{\mathcal{B}} \in \mathcal{D}$ of sufficiently large radius to cover $\mathcal{D}$.
   (c) Let $\mathcal{L}_0 = \mathcal{B}_0$ be the initial list of balls.
   (d) Let $U_0 = \overline{f}(\mathcal{B}_0)$ be the initial upper bound for $\min_{x \in \mathcal{B}_0} f(x)$.
   (e) Let $L_0 = \underline{f}(\mathcal{B}_0)$ be the initial lower bound for $\min_{x \in \mathcal{B}_0} f(x)$.
1. While $U_k - L_k > \epsilon$, repeat the following procedure:
   (a) Remove from $\mathcal{L}_k$ balls $\mathcal{B} \in \mathcal{L}_k$ such that $\underline{f}(\mathcal{B}) > U_k$.
   (b) Choose $\mathcal{B} \in \mathcal{L}_k$ such that $\underline{f}(B) = L_k$.
   (c) Split $\mathcal{B}$ into 3^n overlapping subballs $\mathcal{B}_1, ..., \mathcal{B}_{3^n}$ according to the splitting rule in Section 3.1.2 and discard any subballs which lie entirely outside of $\mathcal{D}$. Let $\mathcal{R}_k$ denote the list of remaining subballs and let $\mathcal{L}_{k+1} := (\mathcal{L}_k \setminus \{\mathcal{B}\}) \cup \mathcal{R}_k$.
   (d) Set $U_{k+1} := \min_{\mathcal{B} \in \mathcal{L}_{k+1}} \overline{f}(\mathcal{B})$.
   (e) Set $L_{k+1} := \min_{\mathcal{B} \in \mathcal{L}_{k+1}} \underline{f}(\mathcal{B})$.
   (f) Set $k = k + 1$.
2. Return U_k as the estimate of the global minimum of $f(x)$ over $\mathcal{D}$.

Fowkes et al. [25] proved that, under suitable assumptions, Algorithm 1

converges in a finite number of iterations to within a tolerance $\epsilon > 0$ of the global minimum of $f(x)$.

3.1.1 Calculating Upper Bound: Discarding Balls and Feasible Points

Given a ball $\mathcal{B}$, upper bound $U(\mathcal{B})$ is computed by evaluating $f(x)$ at feasible point in $\mathcal{B}$. Furthermore, Algorithm 1 discards balls $\mathcal{B}$ which lie entirely outside of $\mathcal{D}$ (see Figure 3.1). Since $\mathcal{D}$ is a convex set this is easy to check, indeed the convex programming problem

$$\min_{x \in \mathbb{R}^n} \|x - x_{\mathcal{B}}\|^2$$
$$x \in \mathcal{D}$$

provides a feasible minimiser if the minimum is smaller than $r_{\mathcal{B}}^2$, therefore, if the minimum of problem is larger than $r_{\mathcal{B}}^2$, we know that the ball $\mathcal{B}$ lies entirely outside of $\mathcal{D}$ and can be discarded.

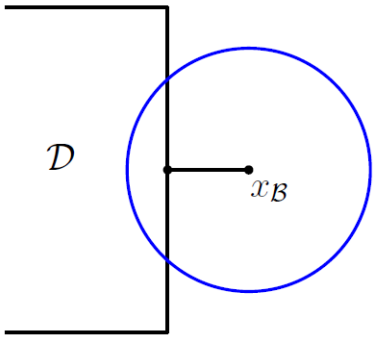

Figure 3.1: Procedure to calculate upper bound and discard balls.

3.1.2 Splitting Rule

Given an initial ball $\mathcal{B}$ covering the domain $\mathcal{D}$ as in Figure 3.2, the splitting rule used in Algorithm 1 proceeds as follows. Each ball $\mathcal{B}$ of radius $r_{\mathcal{B}}$ is split into 3^n overlapping subballs of half-radius $r_{\mathcal{B}}/2$ centred at the vertices of a hypercubic tessellation of edge-length $r_{\mathcal{B}}/\sqrt{n}$ around $x_{\mathcal{B}}$. Formally, each ball $\mathcal{B}$ is split into 3^n overlapping subballs where each subball $\mathcal{B}_i$ of radius $r_{\mathcal{B}}/2$, is centred at

$$x_{\mathcal{B}_i} = x_{\mathcal{B}} + \rho_i^n \left(\frac{-r_{\mathcal{B}}}{\sqrt{2}}, 0, \frac{r_{\mathcal{B}}}{\sqrt{2}} \right)$$

for $i = 1, ..., 3^n$, where $\rho_i^n(s_1, s_2, s_3)$ is a vector in $\mathbb{R}^n$ whose elements are the i-th permutation of s_1, s_2, s_3 taken n at a time with repetition.

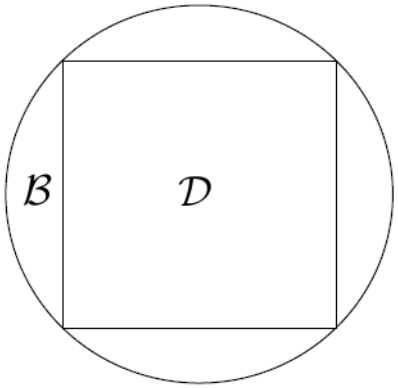

Figure 3.2: Initial covering of domain $\mathcal{D}$ with a ball $\mathcal{B}$.

Note that this choice ensures that the subballs entirely cover the original ball with a constant amount of overlap irrespective of the original ball's radius; this means that at any iteration of Algorithm 1 there is always a covering of closed balls of the convex set $\mathcal{D}$. In Figure 3.3 is showed the splitting procedure for the case $n = 2$.

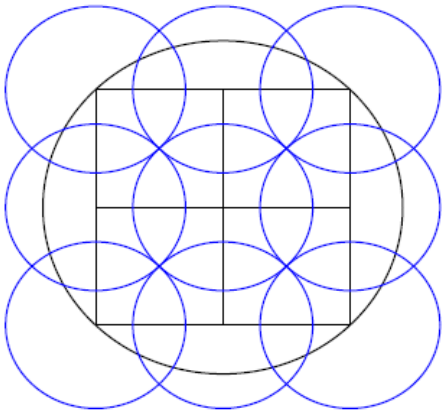

Figure 3.3: An illustration of the splitting rule in two dimensions. The black circle represents the original ball that covers the domain $\mathcal{D}$. This circle is split into nine blue circles of half radius centred at the vertices of the square tessellation.

One can see from Figure 3.3 that there is an intersection between neighbouring balls and thus the minimiser could exist in two of the balls. The rule proceeds in the same way each time the balls are split into subballs, creating coverings with exactly the same amount of overlap on many different levels (see Figure 3.4). In particular, this means that at each level the minimiser

can only exist in at most two balls.

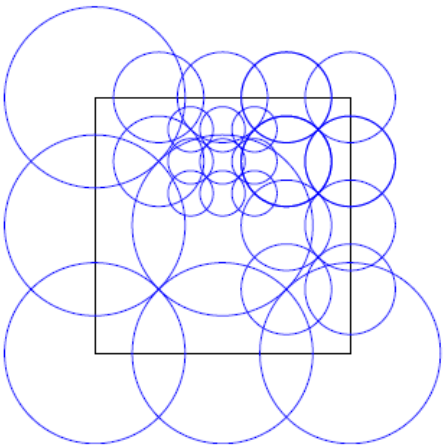

Figure 3.4: An illustration of the coverings of balls at different levels in two dimensions. Note how the overlap between neighbouring balls is preserved across the different levels.

As well as this rule it is possible to use instead a standard rectangular partitioning strategy, where a hyper-rectangle is split into a partition with at most 2^n subrectangles. One can then use a strategy where balls are circumscribed around the hyper-rectangles and apply Algorithm 1 as before. Nevertheless, while this has the advantage of lowering the number of subregions created at each step of the algorithm (in fact the hyper-rectangle is split into a partition with at most 2^n subrectangles), because of non regularity of the hyper-rectangles this creates a large amount of overlap making it difficult to discard uninteresting regions.

3.2 Lipschitz lower bounds Improvements

Cartis et al. [10] propose new bounding techniques using refinements of the bounds (3.2) and (3.3). The new proposals are tested in the oBB framework which allows efficient global solution of the non-convex lower bounding subproblems

$$\min_{x \in \mathcal{B}} l_{\mathcal{B}}(x) \tag{3.4}$$

where $l_{\mathcal{B}}(x)$ is either $q_{\mathcal{B}}(x)$ in (3.2) or $c_{\mathcal{B}}(x)$ in (3.3), over each subdomain $\mathcal{B}$, by letting $\mathcal{B}$ be a Euclidean ball which makes (3.4) tractable. The section starts by looking at improved estimates for the first order lower bound (3.2) and then extends some of these ideas to the second order lower bound (3.3).

3.2.1 First order lower bounds

The approach taken to estimate the gradient Lipschitz constant in Fowkes et al. [25] is to bound the norm of the Hessian over a suitable domain using interval arithmetic. Nonetheless, in the literature there are other approaches which provide suitable estimates for the first order lower bound (3.2). Evtushenko and Posypkin [18, 19] replace the negative Lipschitz constant by a lower bound on the spectrum of the Hessian, $\lambda_{min}(H(x))$, for x in some interval (see Lemma 1 in Appendix B). They approximate $\lambda_{min}(H(x))$ using Gershgorin's Theorem, but other approximations have been proposed in the literature. In particular, we can consider the following possible lower bounds which all require the following lower and upper bounds on the Hessian.

Definition 8. Let Assumption 2 hold. Let $h_{ij}(\xi)$ denote the elements of the Hessian matrix $H(\xi)$ of f. Furthermore, let $\underline{H} = (\underline{h}_{ij})_{1\leq i,j\leq n}$, $\overline{H} = (\overline{h}_{ij})_{1\leq i,j\leq n}$ be such that for all $i, j = 1, ..., n$

$$\underline{h}_{ij} \leq h_{ij}(\xi) \leq \overline{h}_{ij} \tag{3.5}$$

for all ξ in a convex, compact subdomain $\mathcal{B}$.

Theorem 6 (Floudas, 1999 [22])**.** *Let Assumption 2 hold. Given the elementwise bounds $\underline{h}_{ij}, \overline{h}_{ij}$ and corresponding matrices $\underline{H}, \overline{H}$ in (3.5), the following lower bounds for $\lambda^{\mathcal{B}}_{min}(H)$ in the bound (B.1) hold:*

*i) Gershgorin's Theorem (**Ger**):*

$$\lambda^{\mathcal{B}}_{min}(H) \geq \min_i \left[\underline{h}_{ij} - \sum_{j\neq i} \max\left\{ |\underline{h}_{ij}|, |\overline{h}_{ij}| \right\} \right] \tag{3.6}$$

*ii) E-Matrix Diagonal (**Ediag**):*

$$\lambda^{\mathcal{B}}_{min}(H) \geq \lambda_{min}(H_M) - \rho(\Delta H) \tag{3.7}$$

where $\lambda_{min}(H_M)$ denotes the smallest eigenvalue of the midpoint matrix $H_M := \frac{\overline{H}+\underline{H}}{2}$ and $\rho(\Delta H)$ the spectral radius of the radius matrix $\Delta H := \frac{\overline{H}-\underline{H}}{2}$.

*iii) E-Matrix Zero (**E0**):*

$$\lambda^{\mathcal{B}}_{min}(H) \geq \lambda_{min}(\widetilde{H_M}) - \rho(\widetilde{\Delta H}) \tag{3.8}$$

where the modified radius matrix $\widetilde{\Delta H}$ is ΔH with zeros on the diagonal and the modified midpoint matrix $\widetilde{H_M}$ is H_M with $\underline{h}_{ii}$ on the diagonal.

*iv) Lower Bounding Hessian (**lbH**):*

$$\lambda_{min}^{\mathcal{B}}(H) \geq \lambda_{min}(L) \tag{3.9}$$

where the lower bounding Hessian $L = (l_{ij})$ is defined as

$$l_{ij} = \begin{cases} \underline{h}_{ij} + \sum_{k \neq i} \frac{\underline{h}_{ik} - \overline{h}_{ik}}{2} & \text{if } i = j \\ \frac{\underline{h}_{ij} + \overline{h}_{ij}}{2} & \text{if } i \neq j \end{cases}$$

*v) Hertz's Method (**Hz**):*

$$\lambda_{min}^{\mathcal{B}} = \min_k \{\lambda_{min}(H_k)\} \tag{3.10}$$

where the vertex matrices H_k are defined as follows: Let $x \in \mathbb{R}^n$, then there are 2^{n-1} possible combinations for the signs of the $x_i x_j$ products $(i \neq j)$. For the k-th such combination, define the vertex matrix $H_k = (h_{ij}^k)$ where

$$h_{ij}^k = \begin{cases} \underline{h}_{ii} & \text{if } i = j \\ \underline{h}_{ij} & \text{if } x_i x_j \geq 0, i \neq j \\ \overline{h}_{ij} & \text{if } x_i x_j < 0, i \neq j \end{cases}$$

Proof. See Floudas (1999, Section 12.4) for proofs of the above lower bounds (3.6)–(3.10). □

Returning to the gradient Lipschitz constant we can consider the following lower bound on the best $-L_g(\mathcal{B})$ in (3.2).

Theorem 7 (Norm of the Hessian (**Norm**))**.** *Let Assumption 2 hold. Suppose $\mathcal{B} \subset \mathcal{C}$ is a convex, compact subdomain and $x_{\mathcal{B}} \in \mathcal{B}$. Then, for any $x \in \mathcal{B}$, the first order lower bound (3.2) holds. Furthermore, a lower bound for the best $-L_g(\mathcal{B})$ in (3.2) is given by*

$$-L_g(\mathcal{B}) \geq -\sqrt{\sum_{ij} \max\{|\underline{h}_{ij}|, |\overline{h}_{ij}|\}^2} \tag{3.11}$$

where the elementwise bounds $\underline{h}_{ij}, \overline{h}_{ij}$ are defined in (3.5).

Proof. Recalling that $\|M\|_2 \leq \|M\|_F$ for any matrix M where

$$\|M\|_F := \sum_i \sum_j M_{ij},$$

we have from Taylor's theorem to first order and Cauchy-Schwarz that for any $x, y \in \mathcal{B}$

$$\begin{aligned}
\|g(x) - g(y)\|_2 &\leq \left\| \int_0^1 H(y + \tau(x-y))(x-y) d\tau \right\|_2 \\
&\leq \max_{0\leq\tau\leq 1} \|H(y + \tau(x-y))\|_2 \|x-y\|_2 \\
&\leq \max_{0\leq\tau\leq 1} \|H(y + \tau(x-y))\|_F \|x-y\|_2 \\
&= \max_{0\leq\tau\leq 1} \left(\sum_{ij} [H(y + \tau(x-y))]_{ij}^2 \right)^{1/2} \|x-y\|_2 \\
&\leq \left(\sum_{ij} \max\left\{ |\underline{h}_{ij}|^2, |\overline{h}_{ij}|^2 \right\} \right)^{1/2} \|x-y\|_2 \\
&= \left(\sum_{ij} \max\left\{ |\underline{h}_{ij}|, |\overline{h}_{ij}| \right\}^2 \right)^{1/2} \|x-y\|_2.
\end{aligned}$$

Thus the gradient g is Lipschitz continuous on a compact domain $\mathcal{B}$ with l^2-norm Lipschitz constant $\sqrt{\sum_{ij} \max\{|\underline{h}_{ij}|, |\overline{h}_{ij}|\}^2}$. In particular, this means that for the best gradient Lipschitz constant $L_g(\mathcal{B})$, we have for all $x \in \mathcal{B}$

$$L_g(\mathcal{B}) \leq \sqrt{\sum_{ij} \max\{|\underline{h}_{ij}|, |\overline{h}_{ij}|\}^2}.$$

□

3.2.2 Second order lower bounds

In the previous section we have seen how we can replace the gradient Lipschitz constant in (3.2) by an estimate of the smallest eigenvalue of the Hessian. An extension of this approach is also possible for the second order lower bound (3.3) by replacing the Hessian Lipschitz constant L_H with an estimate of the smallest eigenvalue of the derivative tensor $\lambda_{min}^{l^3}(T(x))$; Appendix C introduces some tensor eigenvalue notation and shows why lower bounds on the spectrum of the derivative tensor can be used in place of L_H in (3.3).

Some of the approaches seen in Section 3.2.1 to obtain lower bounds on the smallest eigenvalue in the case of a Hessian matrix can be generalized to the case of a third order derivative tensor. Even though there are l^3-eigenvalue algorithms that are guaranteed to converge to the smallest eigenvalue, these are only applicable to tensors with non-negative (or equivalently

non-positive) entries. The tensor generalisations of the matrices required for the lower bounding strategies presented in (3.7)–(3.10), have, in general, both positive and negative entries, while the generalisation of Gershgorin's Theorem does not require an eigenvalue algorithm and we can therefore generalise Theorem 6 (3.6) to tensors. Given the following definition we can give the generalised theorem.

Definition 9. Let Assumption 3 hold. Let $t_{ijk}(\xi)$ denote the elements of the third order derivative tensor $T(\xi)$. Furthermore, let $\underline{T} = (\underline{t}_{ijk})_{1\leq i,j,k\leq n}$, $\overline{T} = (\overline{t}_{ijk})_{1\leq i,j,k\leq n}$ be such that for all $i, j, k = 1, ..., n$

$$\underline{t}_{ijk} \leq t_{ijk}(\xi) \leq \overline{t}_{ijk} \tag{3.12}$$

for all ξ in a convex, compact subdomain $\mathcal{B}$.

Theorem 8 (Gershgorin's Theorem for the derivative Tensor (**Ger T**))**.** *Let Assumption 3 hold. Assuming the elementwise bounds $\underline{t}_{ijk}, \overline{t}_{ijk}$ in (3.12), $\lambda_{min}^{l^3,\mathcal{B}}(T)$ in (C.2) can be bounded below by*

$$\lambda_{min}^{l^3,\mathcal{B}}(T) \geq \min_i \left[\underline{t}_{iii} - \sum_{k\neq j\neq i} \max\left\{ |\underline{t}_{ijk}|, |\overline{t}_{ijk}| \right\} \right]. \tag{3.13}$$

Proof. Let $\xi \in \mathcal{B}$ be arbitrary. We have from Qi, 2005 [57] that Gershgorin's Theorem for tensors applied to the third order derivative tensor $T(\xi)$ gives

$$\begin{aligned} \lambda_{min}^{l^3}(T(\xi)) &= \min_i \left[\underline{t}_{iii} - \sum_{k\neq j\neq i} |t_{ijk}(\xi)| \right] \\ &\geq \min_i \left[\underline{t}_{iii} - \sum_{k\neq j\neq i} \max\left\{ |\underline{t}_{ijk}|, |\overline{t}_{ijk}| \right\} \right] \end{aligned}$$

for any $\xi \in \mathcal{B}$. As $\lambda_{min}^{l^3,\mathcal{B}}(T) = \min_{\xi\in B} \lambda_{min}^{l^3}(T(\xi))$ from (C.3), the result follows. □

Additionally, an extension of the Norm bound (3.11) from Theorem 7 can be used as a bound on the Hessian Lipschitz constant in (3.3).

Theorem 9 (Norm of the derivative tensor (**Norm T**))**.** *Let Assumption 3 hold. Suppose $\mathcal{B} \subset \mathcal{C}$ is a convex, compact subdomain and $x_{\mathcal{B}} \in \mathcal{B}$. Then, for any $x \in \mathcal{B}$, the second order lower bound (3.3) holds. Furthermore, a lower bound for the best $-L_H(\mathcal{B})$ in (3.3) is given by*

$$-L_H(\mathcal{B}) \geq -\sqrt{\sum_{ijk} \max\left\{ |\underline{t}_{ijk}|, |\overline{t}_{ijk}| \right\}^2} \tag{3.14}$$

where the elementwise bounds $\underline{t}_{ijk}, \overline{t}_{ijk}$ are defined as in (3.12).

Proof. As in the matrix case we have that $\|T\|_2 \leq \|T\|_F$ for any tensor T (see Lemma 6.1 in Fowkes et al., 2013 [25], for a proof). We have from Taylor's theorem to first order and Cauchy-Schwarz that for any $x, y \in \mathcal{B}$

$$\begin{aligned}\|H(x) - H(y)\|_2 &\leq \left\| \int_0^1 T(y + \tau(x-y))(x-y)d\tau \right\|_2 \\ &\leq \max_{0\leq\tau\leq 1} \|T(y+\tau(x-y))\|_F \|x-y\|_2 \\ &= \max_{0\leq\tau\leq 1} \left(\sum_{ijk} \max\left\{|\underline{t}_{ijk}|, |\bar{t}_{ijk}|\right\}^2 \right)^{1/2} \|x-y\|_2.\end{aligned}$$

Thus the Hessian H is Lipschitz continuous on a compact domain $\mathcal{B}$ with l^2-norm Lipschitz constant $\sqrt{\sum_{ijk} \max\left\{|\underline{t}_{ijk}|, |\bar{t}_{ijk}|\right\}^2}$. In particular, this means that for the best Hessian Lipschitz constant $L_H(\mathcal{B})$, we have

$$L_H(\mathcal{B}) \leq \sqrt{\sum_{ijk} \max\left\{|\underline{t}_{ijk}|, |\bar{t}_{ijk}|\right\}^2}.$$

□

3.3 Parallelization of oBB

Similarly to other B&B algorithms, there is also the curse of dimensionality due to the high number of balls in each oBB covering. At each iteration, oBB splits a ball into 3^n smaller subballs while traditional B&B splits a box into only two subboxes. However, the two splitting approaches can be compared if we consider each ball in oBB being split into 3^n subballs, and each box in traditional B&B being split into 2^n subboxes.
Due to the curse of dimensionality, many parallel B&B algorithms over boxes have been proposed in the literature. In particular Gendron and Crainic, 1994 [27] and Crainic et al., 2006 [11] have classified the main approaches into two classes. In Type I there is a parallelization of the operations on subproblems (e.g. bounding) whereas the BB tree is explored in serial (i.e. by one processor). In Type II parallelism by contrast, the tree itself is explored in parallel by many processors. Cartis et al. [10] consider both the types of parallelism of which we give an overview in the following sections.

3.3.1 Data Parallelism: Bounds in Parallel

The idea behind data parallelism is to share the computation of the bounds amongst many processor cores through the implementation of a master/worker approach. The master processor core runs the entire algorithm

except for the operations necessary for obtaining bounds on each subdomain, which are divided amongst itself and the worker processors. Obviously, this approach is useful only if there are many bounding calculations that can be performed independently at the same time and if these calculations are relatively expensive compared to the rest of the algorithm. This parallel variant (see Algorithm 2 below) follows the same line of thought described in Section 3.1, thus, it solves (3.4) to obtain a lower bound $\underline{f}(\mathcal{B})$ on the objective function f over the subdomain $\mathcal{B}$, that is

$$\underline{f}(\mathcal{B}) := \min_{x \in \mathcal{B}} l_{\mathcal{B}}(x) \tag{3.15}$$

where $l_{\mathcal{B}}(x)$ can be either any of the original lower bounds given in (3.2), (3.3) or any of the improved ones defined in (B.1), (C.2). The upper bound $\overline{f}(\mathcal{B})$ on f over $\mathcal{B}$ is simply the objective function f evaluated at a feasible point $x_F \in \mathcal{B}$, that is

$$\overline{f}(\mathcal{B}) := f(x_F). \tag{3.16}$$

It is important to note that if we run this algorithm on one master processor core, we recover the serial version of oBB.

Algorithm 2 : Data Parallel Branch and Bound Algorithm

Master Processor

0. Initialization:
   (a) Set $k = 0$ and t_{max} to be the maximum runtime.
   (b) Let $\mathcal{B}_0$ be a ball with centre $x_{\mathcal{B}} \in \mathcal{D}$ of sufficiently large radius to cover $\mathcal{B}$.
   (c) Let $\mathcal{L}_0 = \mathcal{B}_0$ be the initial list of balls.
   (d) Let $U_0 = \overline{f}(\mathcal{B}_0)$ be the initial upper bound for $\min_{x \in \mathcal{B}_0} f(x)$.
   (e) Let $L_0 = \underline{f}(\mathcal{B}_0)$ be the initial lower bound for $\min_{x \in \mathcal{B}_0} f(x)$.
1. While $U_k - L_k > \epsilon$ and the runtime $< t_{max}$, repeat the following procedure:
   (a) Pruning: Remove from $\mathcal{L}_k$ balls $\mathcal{B} \in \mathcal{L}_k$ such that $\underline{f}(\mathcal{B}) > U_k$.
   (b) Branching: Choose $\mathcal{B} \in \mathcal{L}_k$ such that $\underline{f}(\mathcal{B}) = L_k$.
   (c) Splitting: Split $\mathcal{B}$ into 3^n overlapping subballs $\mathcal{B}_1, ..., \mathcal{B}_{3^n}$ according to the splitting rule in Section 3.1.2 and discard any subballs which lie entirely outside of $\mathcal{D}$. Let $\mathcal{R}_k$ denote the list of remaining subballs and let $\mathcal{L}_{k+1} := (\mathcal{L}_k \setminus \{\mathcal{B}\}) \cup \mathcal{R}_k$.
   (d) Bounding: Partition $\mathcal{R}_k$ into P subsets $\mathcal{R}_k^p$ for $p \in \{1, ..., P\}$ and distribute them amongst the P worker processors for bounding. Wait until all the bounds $\underline{f}(\mathcal{B}), \overline{f}(\mathcal{B})$ for $\mathcal{B} \in \mathcal{R}_k$ are received back.
   (e) Set $U_{k+1} := \min_{\mathcal{B} \in \mathcal{L}_{k+1}} \overline{f}(\mathcal{B})$.
   (f) Set $L_{k+1} := \min_{\mathcal{B} \in \mathcal{L}_{k+1}} \underline{f}(\mathcal{B})$.
   (g) Set $k = k + 1$.
2. Send termination signal to worker processors.
3. Return U_k as the estimate of the global minimum of $f(x)$ over $\mathcal{D}$.

Worker Processor p

1. Repeat the following procedure until termination signal is received:
   (a) Wait for a set of balls $\mathcal{R}_k^p$ from the master processor.
   (b) When the set is received, calculate bounds $\underline{f}(\mathcal{B}), \overline{f}(\mathcal{B})$ for each ball $\mathcal{B} \in \mathcal{R}_k^p$ and send the bounds back to the master processor.

3.3.2 Task Parallelism: Tree in Parallel

In this kind of parallelism the focus is on exploring the BB tree in parallel using several processor cores that generate different sections of the tree starting from different subregions (see Figure 3.5). Unlike rectangular partition where each subregion forms a distinct partition of the domain and any subregions split from it are also contained within that partition, using overlapping balls the parallelisation is more difficult since the balls do not form natural partitions, thus, several processor cores can end up bounding and splitting the same promising ball, doubling the work. The solution

proposed by Cartis et al. [10] to this problem is to eliminate the doubling entirely through efficient communication of centres and radii from the workers to the master. To do this, the master processor keeps a list of all the balls created so far and any new balls created by the worker processors are cross-checked against this list to see if they already exist. However, sending the centre and radius of each ball would be prohibitively expensive. The cost of communication can be greatly reduced if we instead send an integer hash (Knuth, 1998 [34], Section 6.4) of each centre and radius. In particular, we only need to send a hash of one radius and at most 3^n balls since every time a worker processor splits a ball it needs to check whether at most 3^n balls of the same radius exist.[1] Another performance improvement implemented is the use of a priority queue (i.e., an ordered list) to store the subproblems. In particular, the list of balls is ordered according to the lower bound $\underline{f}(\mathcal{B})$, with the smallest lower bound included first in the list. Since most modern HPC clusters consists of a large number of nodes (i.e. sets of processors which share the same memory) interconnected by gigabit ethernet or infiniband switches, it has also implemented a two tier strategy to balance the load between processor cores, i.e. the number of balls, or equivalently the number of subproblems, on each processor core. This strategy allows to load balance both within each node where communication via shared memory will be very efficient and across different nodes where communication via gigabit ethernet or infiniband will be relatively slow. The complete task parallel B&B algorithm is given below, with the lower and upper bounds calculated as before in (3.15),(3.16). Details of the hashing and load balancing are presented later in Sections 3.3.3 and 3.3.4.

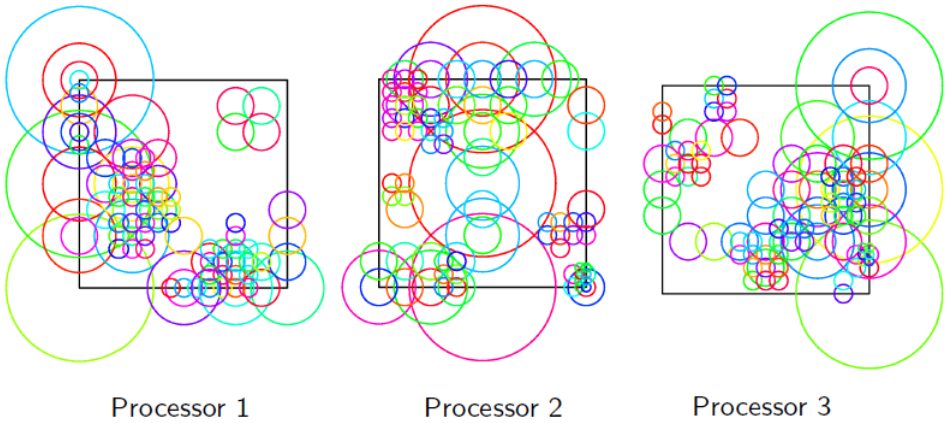

Figure 3.5: Graphical Results of the master/slave Task Parallel implementation of oBB.

[1]Note that the hashes are not guaranteed to be unique and there is a chance that the algorithm will occasionally discard a ball that does not already exist. This extremely rare event can be corrected by running a local solver at the end of the algorithm.

Algorithm 3 : Task Parallel Branch and Bound Algorithm

Master Processor

0. Initialization:
   (a) Set t_{max} to be the maximum runtime of the algorithm.
   (b) Let $\mathcal{B}$ be a ball with centre $x_{\mathcal{B}} \in \mathcal{D}$ of sufficiently large radius to cover $\mathcal{D}$.
   (c) Split $\mathcal{B}$ into 3^n overlapping subballs according to the splitting rule in Section 3.1.2 and discard any subballs that lie entirely outside of $\mathcal{B}$. Partition the remaining subballs into P subsets and distribute them amongst the p worker processors as sets $\mathcal{L}^p$ for $p \in \{1, ..., P\}$.
   (d) Let $\mathcal{R} = \emptyset$ be the initial ordered list of hashes of radii.
   (e) Let $\mathcal{C} = \emptyset$ be the initial ordered list of sets of hashes of centres with the same radius.
1. While $\mathcal{L}^p \neq \emptyset \; \forall p$ and the runtime $< t_{max}$, repeat the following procedure:
   (a) Asynchronously receive U_p and the size $|\mathcal{L}^p|$ of the set $\mathcal{L}^p$ from all $p \in \{1, ..., P\}$ worker processors.
   (b) Asynchronously send $U := \min_{p \in \{1,...,P\}} U_p$ to all P worker processors.
   (c) Hashing: Process lists of hashes received from worker processors, updating $\mathcal{R}$, the list of radius hashes,[2]and $\mathcal{C}$, the list of ball-centre hashes, and inform the workers of any duplicate entries (see Section 3.3.3).
   (d) Perform load balancing across nodes (see Section 3.3.4).
   (e) Perform load balancing within nodes (see Section 3.3.4).
2. Send termination signal to worker processors.
3. Return U as the estimate of the global minimum of $f(x)$ over $\mathcal{D}$.

Worker Processor p

1. Initialization
   (a) Receive workload $\mathcal{L}^p$ from master processor.
   (b) Calculate bounds $\underline{f}(\mathcal{B})$ and $\overline{f}(\mathcal{B})$ as defined in (3.15) and (3.16), respectively, for each ball $\mathcal{B} \in \mathcal{L}^p$ and convert $\mathcal{L}^p$ into a priority queue w.r.t. $\underline{f}(\mathcal{B})$.
   (c) Set $U_p := \min_{\mathcal{B} \in \mathcal{L}^p} \overline{f}(\mathcal{B})$.
   (d) Asynchronously send U_p and $|\mathcal{L}^p|$ to master processor.
   (e) Asynchronously receive U from master processor.
2. Repeat the following procedure until termination signal is received:
   (a) Pruning: Remove from the priority queue $\mathcal{L}^p$ balls $\mathcal{B}$ such that $\underline{f}(\mathcal{B}) > U - \epsilon$.

(b) Branching: Let $\mathcal{B}$ be the first element in the priority queue $\mathcal{L}^p$.[3] Split $\mathcal{B}$ into 3^n overlapping subballs overlapping subballs according to the splitting rule in Section 3.1.2 and discard any subballs that lie entirely outside of $\mathcal{D}$. Let $\mathcal{R}$ denote the list of remaining subballs.

(c) Hashing: Generate an integer hash for each ball in $\mathcal{R}$ and an integer hash for the radius. Send the integer hashes to master processor to see if any of the balls already exist. (Synchronised hashing only: Start bounding $f(x)$ for each ball in $\mathcal{R}$ until the master processor sends the results of the check back). Receive an ordered integer list from the master processor that contains either 1 or 0 depending on whether each ball exists and update $\mathcal{R}$ accordingly.

(d) Bounding: Calculate bounds $\underline{f}(\mathcal{B}), \overline{f}(\mathcal{B})$ according to (3.15), (3.16), for each ball $\mathcal{B} \in \mathcal{R}$ if not already bounded.

(e) Remove the split ball $\mathcal{B}$ from the priority queue $\mathcal{L}^p$ and add the list of remaining subballs $\mathcal{R}$ to $\mathcal{L}^p$.

(f) Set $U_p := \min_{\mathcal{B} \in \mathcal{L}^p} \overline{f}(\mathcal{B})$.

(g) Load Balancing: Asynchronously send the requested number of subproblems from the current workload to the required processor(s) as instructed by the master processor and update $\mathcal{L}^p$ accordingly. If more subproblems are requested than in the current workload, send as many as possible. Send confirmation to the master processor once the send has completed.

(h) Load Balancing: Asynchronously receive subproblems from other processors and update $\mathcal{L}^p$ accordingly.

(i) Asynchronously send U_p and $|\mathcal{L}^p|$ to master processor.

(j) Asynchronously receive U from master processor.

3.3.3 Hashing

In this section we will describe the hashing process used in Algorithm 3 in more detail. In particular, the master processor keeps a list $\mathcal{R}$ containing hashes of the radius and a corresponding list $\mathcal{C}$ of sets of hashes of centres of balls with that radius: for example if $\mathcal{R} = \{\#r_1, \#r_2\}$ and $\mathcal{C} = \{\{\#x_{\mathcal{B}_1}, \#x_{\mathcal{B}_2}\}, \{\#x_{\mathcal{B}_3}\}\}$ then balls $\mathcal{B}_1$, $\mathcal{B}_2$ have radius r_1 and $\mathcal{B}_3$ has radius r_3. Every time a ball is split into subballs the worker has only to send one radius hash and the corresponding centre hashes while the master

[2]Note that $\mathcal{R}$ is always a finite set since the radius is halved each time a ball is split, hence, there can only be a finite number of radii before numerical underflow occurs.

[3]Note that since $\mathcal{L}^p$ is a priority queue w.r.t. $\underline{f}(\mathcal{B})$, $\mathcal{B}$ has the smallest lower bound $\underline{f}(\mathcal{B})$ of all balls in $\mathcal{L}^p$.

processor can quickly determine the radius of the split balls when receives this information. The hash of the radius is simply computed by multiplying for a constant and by converting to a 32-bit integer. For hashing the centre of each ball, it is used a variant of the hash function from Section 4.1 of Teschner et al., 2003 [67], that is, for $x \in \mathbb{R}^n$, given a collection of large primes $p_1, ..., p_n$ and a resolution r, the hash is

$$\#x = \left\lfloor \frac{x_1}{r} \right\rfloor p_1 \veebar \left\lfloor \frac{x_2}{r} \right\rfloor p_2 \veebar \cdots \veebar \left\lfloor \frac{x_n}{r} \right\rfloor p_n$$

where $\veebar$ denotes a bitwise xor (i.e. an exclusive or on the binary digits). Similarly to radius, the hash is then converted to a 32-bit integer which ensures that the communication is as efficient as possible.
Due to performance consideration on the order in which the master processor deals with the incoming hashes there are two different suitable approaches. In the first approach, the master processes the hashes one at a time as they are received and the workers simply wait for confirmation of which balls already exist before bounding them. This approach is suitable in the cases where the balls are inexpensive to bound relative to the cost of communicating the hashes. When, instead, bounding the balls is expensive relative to the communication cost, the workers tend to spend a significant amount of time waiting for a response from the master. The second approach therefore tries to solve these problems by getting the master to process the hashes from all the workers in one go while the workers start bounding the balls in the background. We can now describe how the step 1c in Algorithm 3 works considering either the approaches (one-at-a-time and synchronised).

Algorithm 4 : One-at-a-time Hashing

Master Processor: (Step 1c of Algorithm 3)

1. If a worker processor p wants to check if a set of balls of the same radius already exists, receive a list containing an integer hash $\#c_i$ of the centre of each ball $\mathcal{B}_i$ and an integer hash $\#r$ of the radius.

2. Check if $\#r$ is in $\mathcal{R}$. If it is not, append $\#r$ to $\mathcal{R}$, append the set of $\#c_i$'s to $\mathcal{C}$ since they cannot already be present in $\mathcal{C}$. If $\#r$ is in $\mathcal{R}$, check if any of the $\#c_i$'s are already present in the corresponding set in $\mathcal{C}$. Add any $\#c_i$'s that are not present to the corresponding set in $\mathcal{C}$.

3. Set $\mathcal{E}$ to be an ordered list that contains either 1 or 0 for each i depending on whether $\#c_i$ is present in the corresponding set in $\mathcal{C}$ or not and send $\mathcal{E}$ to worker processor p.

Algorithm 5 : Synchronised Hashing

Master Processor: (Step 1c of Algorithm 3)

1. Receive from all worker processors p, a list containing integer hashes of the radius $\#r^p$ and centres $\{\#c_i^p\}$ of each ball $\mathcal{B}_i^p$ on processor p wanting to be checked.

2. For each p, check if $\#r^p$ is in $\mathcal{R}$. If it is not, append $\#r^p$ to $\mathcal{R}$, append the set of $\#c_i^p$'s to $\mathcal{C}$ since they cannot already be present in $\mathcal{C}$. If $\#r^p$ is in $\mathcal{R}$, check if any of the $\#c_i^p$'s are already present in the corresponding set in $\mathcal{C}$. Add any $\#c_i^p$'s that are not present to the corresponding set in $\mathcal{C}$.

3. For each p, set $\mathcal{E}^p$ to be an ordered list that contains either 1 or 0 for each i depending on whether $\#c_i^p$ is present in the corresponding set in $\mathcal{C}$ or not and send $\mathcal{E}^p$ to worker processor p.

3.3.4 Load Balancing

In this section we describe the two load balancing approaches, starting with load balancing across processors within nodes. As we would see this first approach forms the basis for load balancing across nodes. In both the approaches it will be used $\mathcal{N}$ throughout the section to denote a node.

Load balancing across processors within a node

At each load balancing step the master processor takes a snapshot of the load on the node and establishes how many subproblems each processor within that node should have so as to be balanced. It then assigns the shortfall from the processor with the largest load to the one with the smallest and updates the snapshot. The procedure is repeated until all processors in the node have a load that does not differ by more than 10%, that is, for all processors $p_1, p_2 \in \mathcal{N}$

$$\frac{|S^{p^1} - S^{p^2}|}{\max\{\min\{S^{p^1}, S^{p^2}\}, 1\}} > 0.1 \tag{3.17}$$

where S^p denotes the load (i.e. the number of subproblems) on processor p. The scheme is given in more detail below.

Algorithm 6 : Master Processor: (Step 1e of Algorithm 3)

For each node $\mathcal{N}$, repeat the following procedure: Let S^p be a snapshot of the load $|\mathcal{L}^p|$ on each worker processor p in $\mathcal{N}$, i.e. a local copy of the load that we will work with. Let $I = \left\lfloor \sum_{p\in\mathcal{N}} S^p/|\mathcal{N}| \right\rfloor$ be the ideal load on each processor in $\mathcal{N}$. We would like all processor loads to be as close as possible to the ideal load I. Set $k = 0$. While (3.17) holds and $k < |\mathcal{N}|$, repeat the following procedure:

1. Let $S^{p_{min}}$ and $S^{p_{max}}$ be the smallest and largest loads in the node $\mathcal{N}$ on processors p_{min} and p_{max} respectively.
2. Calculate $I - S^{p_{min}}$ as the load we need to add to $S^{p_{min}}$ so that it has ideal load.
3. If any previous send has reached its destination, instruct processor p_{max} to asynchronously send $I - S^{p_{min}}$ subproblems to processor p_{min}.
4. Update snapshots: subtract $I - S^{p_{min}}$ from $S^{p_{max}}$ and add $I - S^{p_{min}}$ to $S^{p_{min}}$ so that the previously smallest load increases to I and the previously largest load decreases to $S^{p_{max}} + S^{p_{min}} - I > S^{p_{min}}$ (unless $S^{p_{max}} = I$ in which case it is already balanced).
5. Set $k = k + 1$.

Load balancing across nodes

For load balancing across nodes it is applied a similar scheme (see Algorithm 7). In this case a fraction of the shortfall is assigned from the processor with the largest load and distributes it as evenly as possible to all processors on the node with the smallest load and the process repeats until all the nodes have a load that does not differ by more than 10%, that is, for all nodes $j_1, j_2 = 1, ..., N$

$$\frac{|T^{j_1} - T^{j_2}|}{\max\{\min\{T^{j_1}, T^{j_2}\}, 1\}} > 0.1 \tag{3.18}$$

where T^j denotes the total load on node $\mathcal{N}_j$ for $j = 1, ..., N$.

Algorithm 7 : Master Processor: (Step 1d of Algorithm 3)

Let S^p be a snapshot of the load $|\mathcal{L}^p|$ on each worker processor $p = 1, ..., P$, i.e. a local copy of the load that we will work with. Let $T_j = \sum_{p \in \mathcal{N}_j} S^p$ denote the total load on each node $\mathcal{N}_j$ for $j = 1, ..., N$. Set $k = 0$. While (3.18) holds and $k < P$, repeat the following procedure:

1. Let $I = \left\lfloor \sum_{j=1}^{N} T^j / N \right\rfloor$ be the ideal node load. We would like all node loads to be as close as possible to the ideal node load I.
2. Let $T^{j_{min}}$ and $T^{j_{max}}$ be the smallest and largest node loads, present on the nodes $\mathcal{N}_{j_{min}}$ and $\mathcal{N}_{j_{max}}$ respectively. Calculate $I - T^{j_{min}}$ as the node load we need to add to node $\mathcal{N}_{j_{min}}$ so that it has ideal node load.
3. Let $S^{p_{min}}$ and $S^{p_{max}}$ be the smallest and largest processor loads on nodes $\mathcal{N}_{j_{min}}$ and $\mathcal{N}_{j_{max}}$ respectively. Ideally, we would like to take $I - T^{j_{min}}$ subproblems from processor p_{max} and distribute them evenly across all processors in node $\mathcal{N}_{j_{min}}$. However, this may deplete processor p_{max} so we lower the amount we take by $S^{p_{min}}$ and do not take more than $[S^{p_{max}}/3]$, where $[\cdot]$ denotes rounding to the nearest integer. This gives the actual amount A to take from p_{max} as

$$A = \begin{cases} \max\{I - T^{j_{min}} - S^{p_{min}}, 0\} & \text{if less than } [S^{p_{max}}/3] \\ [S^{p_{max}}/3] & \text{otherwise.} \end{cases}$$

4. If $A > 0$ and any previous send has reached its destination, instruct processor p_{max} to asynchronously send $[A/|\mathcal{N}_{j_{min}}|]$ subproblems to each of the processors on node $\mathcal{N}_{j_{min}}$.
5. Update snapshots: add $[A/|\mathcal{N}_{j_{min}}|]$ to S^p for all processors p in node $\mathcal{N}_{j_{min}}$ and subtract A from $S^{p_{max}}$ so that the previously smallest node load increases by A and the previously largest node load decreases by A.
6. Recompute the total node load $T^j = \sum_{p \in \mathcal{N}_j} S^p$ on each node $\mathcal{N}_j$ for $j = 1, ..., N$.
7. Set $k = k + 1$.

3.4 Numerical Results

In this section we are only going to briefly outline the results reached by the algorithm by considering the different types of lower bound and the different implementations of parallelization (to find out more about the numerical performance, see Cartis et al. [10]). The first and second order estimation approaches have been tested on test sets of

1. Random polynomials
2. Random radial basis functions (RBFs)

with the aim of checking which estimation approach gives the best oBB performance in terms of runtime.

For what concern random polynomials, the second order lower bounds significantly outperform the first order ones (with the tensor Gershgorin approach clearly superior). For the random RBFs, instead, the first order lower bounding Hessian estimation approach outperforms the second order tensor Gershgorin approach, while the tensor norm approach is the best for random RBFs. This suggests that there is no single first or second order bound that is clearly superior considering different objective functions, and also that, there are instances where first order bounds, which are cheaper to compute, can be competitive with second order ones.
For test the parallel performance of data parallel and task parallel algorithms, it is computed the speedup S_P of the parallel algorithm on P processor cores over the serial defined as

$$S_P = \frac{T_1}{T_P} \tag{3.19}$$

where T_1 is the runtime of the serial algorithm while T_P is the runtime of the parallel algorithm on P processors.
For random polynomials and RBFs it has been run a parallel Python-based MPI implementation of both algorithms 2 and 3. With both the algorithms the average speedup is increased and the task parallel algorithm performs significantly better than the data parallel one. Furthermore, for both random polynomials and RBFs, because of the subproblems are inexpensive to solve relative to the cost of communicating the hashes, one-at-a-time hashing significantly outperforms synchronised hashing.
For a more thorough numerical evaluation, both the parallel algorithms have been run on radial basis function approximations to a selection of 31 problems from the COCONUT benchmark whose dimension varies from 4 to 6 (see Shcherbina et al. [62], for details of the benchmark). In recent years radial basis function (RBF) interpolation has become a well established approximation method for real valued functions. The reason of this choice is that one can cheaply obtain derivatives of the RBF of any order and, in particular, this allows to easily obtain the Hessian Lipschitz constant (see Fowkes et al. [25]). One of the main advantages of using RBFs is that the user merely needs to supply the objective function and a set of points at which it should be evaluated to construct the RBF approximation. Instead, the disadvantage is that the optimum found by the algorithm will only be close to the optimum of the objective function if it is sampled at sufficiently

many points. Once again, it is used a parallel Python-based MPI implementation but this time using the synchronous hashing because of the effort for bounding the subproblems. Moreover it is used the tensor norm approach as it performs better for RBF approximations. Once again, the task parallel algorithm reach better speedup, while the data parallel algorithm performs rather poorly. This is due to the fact that more work takes place in exploring the BB tree and not in bounding subproblems which is parallelised in the data parallel algorithm.

Throughout the following, due to the conclusion described above, we will consider only the case of oBB with the use of tensor norm approach (i.e., second order lower bound) and Task Parallelism (with synchronized hash).

Chapter 4

Domain reduction strategies

Nevertheless the branching and bounding improvements considered in the previous chapter, the approach taken in exam is not enough efficient to solve problems with a large size since it requires a considerable effort to solve the high number of subproblems. Because of the exponential explosion of the subproblems, further improvements are needed with the objective of accelerating the algorithm; for this scope a domain reduction strategy is implemented.

4.1 General description

By regarding that we are going to solve a constrained global optimization problems in minimization form with a general (nonconvex) objective function and a convex, compact set; we might indicate explicitly the lower and upper bounds on the variables (from the boundedness of the feasible region), and so consider a problem of the form

$$\min_{x \in \mathcal{D} \cap \mathcal{R}} f(x), \tag{4.1}$$

where $\mathcal{D} \subseteq \mathbb{R}^n$ is a closed convex set, $f : \mathbb{R}^n \to \mathbb{R}$ and

$$\mathcal{R} = \{x \in \mathbb{R}^n : l_j \leq x_j \leq u_j, j \in \{1, ..., n, \}\}$$

is the box containing $\mathcal{D}$ defined by the bounds of the variables. Note that if these bounds are not explicitly given, they can be derived by minimizing and maximizing each variable over the original feasible region. If we consider a general B&B approach based on branching and bounding steps and we restrict our attention to the computation of the lower bound for the original problem (analogous way for any subregion), first we need an underestimator of f over the box $\mathcal{R}$, i.e. a function $\hat{f}$ such that

$$\hat{f}(x) \leq f(x), \ x \in \mathcal{R}.$$

Once we have an underestimator, a lower bound for (4.1) is computed by solving the following problem

$$\min_{x \in \mathcal{D} \cap \mathcal{R}} \hat{f}(x).$$

If we look at the efficiency of a B&Bd approach, this strongly depends on the fathoming rule. Because of a node is fathomed if its lower bound is larger than the current (global) upper bound, then it becomes important to compute a lower bound which is as high as possible. But how can we improve it? Improving $\hat{f}$ over the current box $\mathcal{R}$ is usually difficult, and also impossible if it is already the convex envelope of f over $\mathcal{R}$. Notwithstanding, because of the dependency of $\hat{f}$ on the box $\mathcal{R}$ (in general), we can try to improve $\hat{f}$ by reducing the box $\mathcal{R}$, instead of trying to improve it by keeping $\mathcal{R}$ fixed. This finally leads us to the main topic of this chapter: domain reduction (DR) strategies.
DR strategies, also called Bounds tightening, as introduced in Section 2.9, are considered optional in a general B&B framework in the sense that the algorithm will, in theory, converge even without them. Note that, event thought these procedures generally speed up the algorithm, depending on how computationally expensive they are, in some cases convergence might be faster without their performance. However, the performance of solution methods for GO problems like B&B methods, can often be strongly enhanced through (DR) strategies, thus, in the great majority of cases, this step is essential to achieve fast convergence. Their importance can be demonstrated by the fact that most of global solvers embed such techniques (e.g., in the software BARON [61]). In the next section a particular subclass of DR that plays an important role is described.

4.2 Range Reduction strategies

Range Reduction (RR) strategies aim at reducing as much as possible the ranges of the variables in such a way that no feasible solution (feasibility-based RR) or optimal solution (optimality-based RR) is lost; in particular, they strengthen existing bounds over the variable of the problem for cutting the feasible region. A RR strategy can be defined as follows. Let $\mathcal{R} = \prod_{i=1}^{n} [l_i, u_i]$ be a box containing the feasible region $\mathcal{D}$ of the nonconvex problem and $\mathcal{A}_n$ be the set of all n-dimensional boxes. A RR strategy can be seen as a function

$$RR_{f,\mathcal{D}} : \mathcal{A}_n \to \mathcal{A}_n$$

satisfying the property

$$RR_{f,\mathcal{D}}(\mathcal{R}) \subseteq \mathcal{R} \;\; \forall \mathcal{R} \in \mathcal{A}_n$$

A valid feasibility-based RR is such that

$$\forall \mathcal{R} \supseteq \mathcal{D} \;:\; RR_{f,\mathcal{D}}(\mathcal{R}) \supseteq \mathcal{D},$$

i.e., the result of a reduction is a box which also contains $\mathcal{D}$ and, therefore, does not remove feasible points.
For optimality-based strategies, assuming that a finite upper bound U is available, a valid optimality based RR satisfies

$$\forall \mathcal{R} \supseteq \{x \in \mathcal{D} \ : \ f(x) \leq U\}, \ \ RR_{f,\mathcal{D}}(\mathcal{R}) \supseteq \{x \in \mathcal{D} \ : \ f(x) \leq U\}$$

i.e., we reduce $\mathcal{R}$ without losing optimal solutions but (possibly) losing feasible ones.

Remark 3. When we consider a feasibility-based reduction, if the domain is convex, by minimizing and maximizing each variable over $\mathcal{D}$ is possible obtain the largest possible reduction for all the variables. For what concerns optimality-based reduction, instead, it is simple define the largest possible reduction but not so easy attain it. In particular, let us consider the three boxes

- $\mathcal{R}$: the box defined by the original ranges of the variables,
- $\mathcal{R}_{DR}$: the box obtained from a optimality-based reduction,
- $\mathcal{R}_{LB}$: the smallest box enclosing all feasible points.

Given any optimality-based reduction, the following relation holds

$$\mathcal{R}_{LB} \subseteq \mathcal{R}_{DR} \subseteq \mathcal{R},$$

therefore, $\mathcal{R}_{LB}$ is a lower limit for the reduction. Caprara et al. [9] present a subclass of GO problems for which a suitably chosen reduction strategy is able to guarantee $\mathcal{R}_{DR} = \mathcal{R}_{LB}$ and prove that the same reduction does not guarantee the same result when we slightly enlarge the considered subclass.

4.3 A customary RR strategy

Given the problem (4.1), for what concern feasibility based RRs, the best we can do to reduce the domain of x_k is to solve the two problems

$$\min / \max \{x_k : x \in \mathcal{D} \cap \mathcal{R}\} \tag{4.2}$$

while for optimality based RRs the tightest reduction is obtained by solving the pair of problems

$$\min / \max \{x_k : x \in \mathcal{D} \cap \mathcal{R}, \ f(x) \leq U\}. \tag{4.3}$$

By considering problems (4.2), these are convex if $\mathcal{D}$ is a convex set. For what concern instead problems (4.3), these can be as difficult as the orginal

problem (4.1) when f is nonconvex, even if $\mathcal{D}$ is convex; therefore, the identification of a valid optimality-based RR might be an hard task. If we consider that any lower/upper bound for the minimization/maximization problem in (4.2) and (4.3), by assuming that $\mathcal{D}$ is a convex set, a valid RR strategy can be found by replacing f with a convex underestimator $\hat{f}$ over $\mathcal{D}$. Therefore, problems (4.3) can be rewritten as

$$\min / \max \left\{ x_k : x \in \mathcal{D} \cap \mathcal{R},\ \hat{f}(x) \leq U \right\}. \tag{4.4}$$

Throughout the rest of the book we will call Standard Range Reduction (SRR) the strategies seen in this section.

Remark 4. The importance of a reduction strategy is not merely due to the reduction of the search space. Within BB algorithms, lower bounds are computed by an underestimator (e.g. the convex one $\hat{f}$). If $\hat{f}$ depends on the ranges of the variables, then it gets sharper (i.e., closer to the objective function f) when the ranges of the variables become tighter. Therefore, each time we improve the ranges of the variables, we also improve the convex underestimator which has the effect of improving the lower bounds at the different nodes of a BB tree. Furthermore, one we obtain a better underestimator, we can use it for a further RR, thus, an obvious way to improve the results of SRR is by iterating it.

4.4 Iterated Standard Range Reduction (ISRR)

This reduction is obtained by iteratively applying SRR until no further range reductions are possible for variable x_k or until this procedure produces significative reduction.

Algorithm 8 : ISRR for variable x_k

Step 0 Set $l_k^0 = l_k$, $u_k^0 = u_k$ and $i = 0$

Step 1 Solve the two programs

$$l_k^{i+1}, u_k^{i+1} = \min / \max \left\{ x_k : \hat{f}_i(x) \leq U, x \in \mathcal{D} \cap \mathcal{R}_i \right\},$$

where

$$\mathcal{R}_i = \left\{ x \in \mathbb{R}^n : l_k^i \leq x_k \leq u_k^i, l_j \leq x_j \leq u_j, j \in \{1...n\} \setminus \{k\} \right\},$$

is the current box, $\hat{f}_i$ is the underestimator of f with respect to $\mathcal{R}_i$, and l_k^{i+1}, u_k^{i+1} represent respectively the new lower and upper bounds of x_k.

Step 2 If $l_k^{i+1} = l_k^i$ and $u_k^{i+1} = u_k^i$ then stop. Otherwise, set $i = i + 1$ and repeat Step 1.

In particular, the procedure can either terminates after a finite number of steps or produces an infinite sequence of lower bounds and upper bounds generating a valid reduction for x_k through the values

$$l_k^{ISRR} = \sup_i l_k^i = \lim_{i\to\infty} l_k^i$$

and

$$u_k^{ISRR} = \inf_i u_k^i = \lim_{i\to\infty} u_k^i.$$

4.5 Iterating domain reduction over all the variables

Any RR strategy can be ideally applied to all the variables. A common practice is to fix some order P of the variables for reducing their ranges and iteratively repeat the procedure until the range of at least one variable is tightened. The procedure is the following

Algorithm 9 : Multiple range reduction (MRR)

Step 0 Let $P = \{x_{j_1}, ..., x_{j_n}\}$ be a given order of the variables. Set $l_j^0 = l_j, u_j^0 = u_j$ for $j \in \{0...n\}$ and $i = 0$.

Step 1 For $k = 1, ..., n$ set

$$\left[l_{i_k}^{i+1}, u_{i_k}^{i+1}\right] = range\ reduction(x_{i_k}; \left[l_{i_j}^{i+1}, u_{i_j}^{i+1}\right],$$

$$j = 1, ..., k-1, \left[l_{i_j}^i, u_{i_j}^i\right], j = k, ..., n).$$

Step 2 If $l_j^{i+1} = l_j^i$ and $u_j^{i+1} = u_j^i$ for $j \in \{0, ..., n\}$ then stop. Otherwise, set $i = i + 1$ and repeat Step 1.

Remark 5. If we remove the optimality constraint in Algorithm 8 and 9 we obtain respectively Feasibility-based ISRR and Feasibility-based MRR; in this case in both algorithms the Step 1 is performed just one time because we have not the convex underestimator $\hat{f}$ and thus the possibility of having further reductions through his improvement.

For what concern MRR strategy, given the monotony assumption

Assumption 1. The RR procedure is monotone, i.e. given domains $\left[\tilde{l}_j, \tilde{u}_j\right]$ and $[l_j, u_j]$ that satisfy

$$\left[\tilde{l}_j, \tilde{u}_j\right] \subseteq [l_j, u_j], j = 1, ..., n,$$

we have that, for any variable x_k,

$$RR(x_k; \left[\tilde{l}_j, \tilde{u}_j\right], j = 1, ..., n) \subseteq RR(x_k; [l_j, u_j], j = 1, ..., n).$$

we have the following result

Proposition 4. *Under Assumption 1, the domains computed by MRR satisfy:*

$$l_j^i \to \bar{l}_j \text{ and } u_j^i \to \overline{u}_j asi \to \infty, j = 1, ..., n,$$

and the limits $\bar{l}_j$ and $\overline{u}_j$ do not depend on the order P of the variables.

Proof. See Caprara and Locatelli [8](section 7) or Locatelli and Schoen [39] (page 345) for the proof. □

4.6 Underestimating a general nonconvex function: The α-BB approaches

In the previous chapters we introduced several optimality-based RR strategies which exploit a convex underestimator in order to have a convex problem. How to compute such convex underestimator?
While there exist several strategies for which is required that the objective function and the constrains have to be of special forms (e.g. quadratic or polynomial); the α-BB approaches (see Floudas [22]; Floudas, Androulakis, and Maranas [23]) allow of underestimating a general nonconvex function and the requirements are loosen. We only assume that the function f is twice-continuously differentiable, i.e., $f(x) \in \mathcal{C}^2$. Given a box $\mathcal{R} = \prod_{i=1}^n [l_i, u_i]$, a convex underestimate on $\mathcal{R}$ can be defined as

$$\hat{f}(x) = f(x) - q(x; \alpha) = f(x) - \sum_{i=1}^{n} \alpha_i (x_i - l_i)(u_i - x_i) \tag{4.5}$$

where $\alpha_i \geq 0, i = 1, ..., n$, are parameters such that $q(x) \geq 0$ is satisfied over $\mathcal{R}$. $\hat{f}$ is convex if and only if $\nabla^2 \hat{f}(x)$ is positive semi-definite where

$$\nabla^2 \hat{f}(x) = \nabla^2 f(x) + 2diag(\alpha), \tag{4.6}$$

hence, $\hat{f}$ is convex if large enough values of α_i are chosen. How to choose α_i's?
One possibility is to consider a uniform choice, i.e., $\alpha = \alpha_i$. In this case $\hat{f}$ is convex iff

$$\alpha \geq \max\left\{0, -\frac{1}{2} \min_{x \in [l,u]} \lambda_{min}(x)\right\} \tag{4.7}$$

where $\lambda_{min}(x)$ is the minimum eigenvalue of $\nabla^2 f(x)$. If, instead, we want an nonuniform choice, suitable values are computed by exploiting Gerschgorin theorem. In fact, every eigenvalue of a symmetric matrix A belongs to the interval

$$[A_{ii} - \sum_{i \neq j} |A_{ij}|, A_{ii} + \sum_{i \neq j} |A_{ij}|]$$

Thus, the eigenvalues of $\nabla^2 \hat{f}$ belong to

$$\left[2\alpha_i + \nabla^2_{ii} f(x) - \sum_{j \neq i} |\nabla^2_{ij} f(x)|,\ 2\alpha_i + \nabla^2_{ii} f(x) + \sum_{j \neq i} |\nabla^2_{ij} f(x)| \right]$$

If lower and upper bounds $\underline{H}_{ij}, \overline{H}_{ij}$ for the hessian are known, then it is enough to choose

$$\alpha_i \geq \frac{1}{2} \max\{0; -\underline{H}_{ii} + \sum_{i \neq j} \max\{|\underline{H}_{ij}|, |\overline{H}_{ij}|\}\} \tag{4.8}$$

for any $i = 1, ..., n$. Other possible choices are based on the scaled Gerschgorin theorem: given a positive vector d, $\hat{f}$ is convex if for each $i \in \{1, ..., n\}$

$$\alpha_i \geq \frac{1}{2} \max\{0; -\underline{H}_{ii} + \sum_{i \neq j} \max\{|\underline{H}_{ij}|, |\overline{H}_{ij}| \frac{d_j}{d_i}\}\} \tag{4.9}$$

where possible choices for d are: (i) $d_i = 1\ \forall i = 1, ..., n$ (4.8) or (ii) $d_i = u_i - l_i$ which permit to consider the different scaling of the variables.

Remark 6. The minimization problem which appears in equation (4.7) can be written as

$$\begin{gathered} \min_{x,\lambda} \lambda \\ \nabla^2 f(x) - \lambda I = 0 \\ x \in [l, u] \end{gathered}$$

where I is the identity matrix. This problem, in general, can be a difficult nonconvex optimization problem. A lower bound on the smallest eigenvalue of $\nabla^2 f(x)$ can be used in its place. A valid lower bound can be easily computed when an interval Hessian matrix is introduced. For this scope the lower bounds (3.6)-(3.10) can be used. In addition to these, further method to compute a lower bound for the smallest eigenvalue can be found in Adjiman et al. [1].

Note that the α_i values computed through interval arithmetic might result larger than needed lowering the quality of $\hat{f}$. Moreover, the maximum separation between f and $\hat{f}$ computed as

$$\max(f(x) - \hat{f}(x)) = \frac{1}{4}\sum_{i=1}^{n} \alpha_i \left(\frac{u_i - l_i}{2}\right)^2,$$

suggests that the error in underestimating decreases when the box is split (decreases to 0 if we reduce the box to a single point). For these reasons a spline-based variant has been proposed in Meyer and Floudas [43]. The basic idea of this approach, that follows the concepts treated in Remark **??**, is that of partitioning the original box into many different subboxes over which we can get sharper underestimators. The underestimators over the different regions are then joined to form a single function in the whole domain that satisfies the requirements of convexity, continuity and smoothness. Formally, given a box $\mathcal{R}$, each interval $[l_i, u_i]$ is split into N_i subintervals $[a_i^k, a_i^{k+1}], k = 0, ..., N_i - 1$ where

$$a_i^0 = l_i < a_i^1 < ... < a_i^{N_i - 1} = u_i.$$

For what concern q, this is defined as the sum of a number of terms, equal to the number of variables x_i, that depend of where each variable falls, i.e.,

$$q(x; \alpha) = \sum_{i=1}^{n} q_i^{k_i}(x_i, \alpha_i^{k_i}), \quad x_i \in [a_i^{k_i - 1}, a_i^{k_i}]$$

where

$$\alpha = (\alpha_1, ..., \alpha_n), \quad \alpha_i = (\alpha_i^1, ..., \alpha_i^{N_i}), \quad i = 1, ..., n,$$

and

$$q_i^{k_i}(x_i; \alpha_i^{k_i}) = \alpha_i^{k_i}(x_i - a_i^{k_i - 1})(a_i^{k_i} - x_i) + \beta_i^{k_i} x_i + \gamma_i^{k_i}$$

for each $k_i = 1, ..., N_i$. The values $\alpha_i^{k_i} \geq 0$ are chosen in such a way that $\hat{f}$ is convex over the box $\prod_{i=1}^{n} \left[a_i^{k_i - 1}, a_i^{k_i}\right]$ (e.g., by using (4.8) or (4.9)), while the other parameters are chosen in such a way that q is continuous and smooth. For the choice of $\beta_i^{k_i}$ and $\gamma_i k_i$ we need to impose for each $i = 1, ..., n$ that:

- $q(x, \alpha) = 0$ at the vertices of $\mathcal{R}$;
- continuity at the extremes of each subintervals;
- continuity of the derivatives at the extremes of each subintervals.

Such conditions result in a liner system with $2N_i$ variables and $2N_i$ equations (where the values $\alpha_i^{k_i}$ are supposed to be already computed)

$$\begin{cases} q_i^1(l_i; \alpha_i^1) = q_i^{N_i}(u_i; \alpha_i^{N_i}) = 0 \\ q_i^{k_i}(a_i^{k_i}; \alpha_i^{k_i}) = q_i^{k_i+1}(a_i^{k_i}; \alpha_i^{k_i+1}), \qquad k_i = 1, ..., N_i - 1 \\ \frac{d\, q_i^{k_i}}{d\, x_i}(a_i^{k_i}; \alpha_i^{k_i}) = \frac{d\, q_i^{k_i+1}}{d\, x_i}(a_i^{k_i}; \alpha_i^{k_i+1}), \quad k_i = 1, ..., N_i - 1 \end{cases}$$

The system above can be rewritten as

$$\begin{cases} \beta_i^1 a_i^0 + \gamma_i^1 & = 0 \\ \beta_i^{k_i} a_i^{k_i} + \gamma_i^{k_i} & = \beta_i^{k_i+1} a_i^{k_i} + \gamma_i^{k_i+1} \qquad k_i = 1, ..., N_i - 1 \\ \beta_i^{N_i} a_i^0 + \gamma_i^{N_i} & = 0 \\ -\alpha_i^{k_i}(a_i^{k_i} - a_i^{k_i-1}) + \beta_i^{k_i} = \alpha_i^{k_i+1}(a_i^{k_i+1} - a_i^{k_i}) + \beta_i^{k_i+1} & \quad k_i = 1, ..., N_i - 1. \end{cases}$$

The following example shows the difference between classical and spline-based α-BB (we don't illustrate the numerical results for the parameters). The complete resolution procedure can be found in Meyer and Floudas [43] or Locatelli and Schoen [39].

Example 1. Let

$$f(x) = -2x + 10x^2 - 3x^3 - 5x^4$$

with $\mathcal{R} = [0, 1]$. In Figure the graph of $f(x)$ and $\hat{f}(x)$ computed with spline α-BB are shown and compared with the underestimator of the classic α-BB.

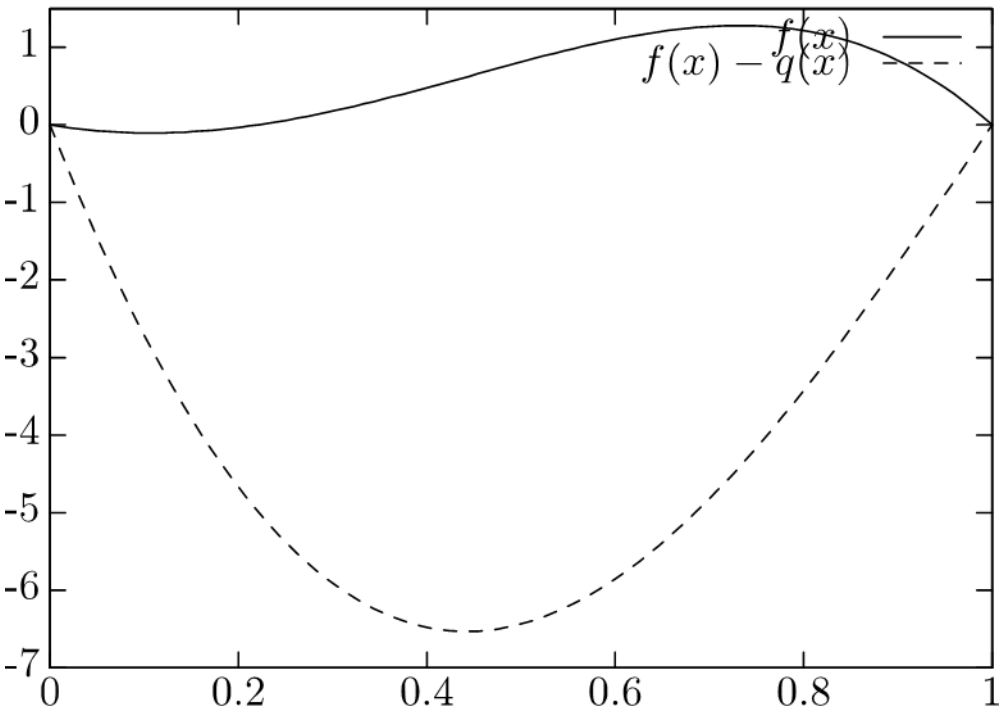

Figure 4.1: Classical α-BB underestimator.

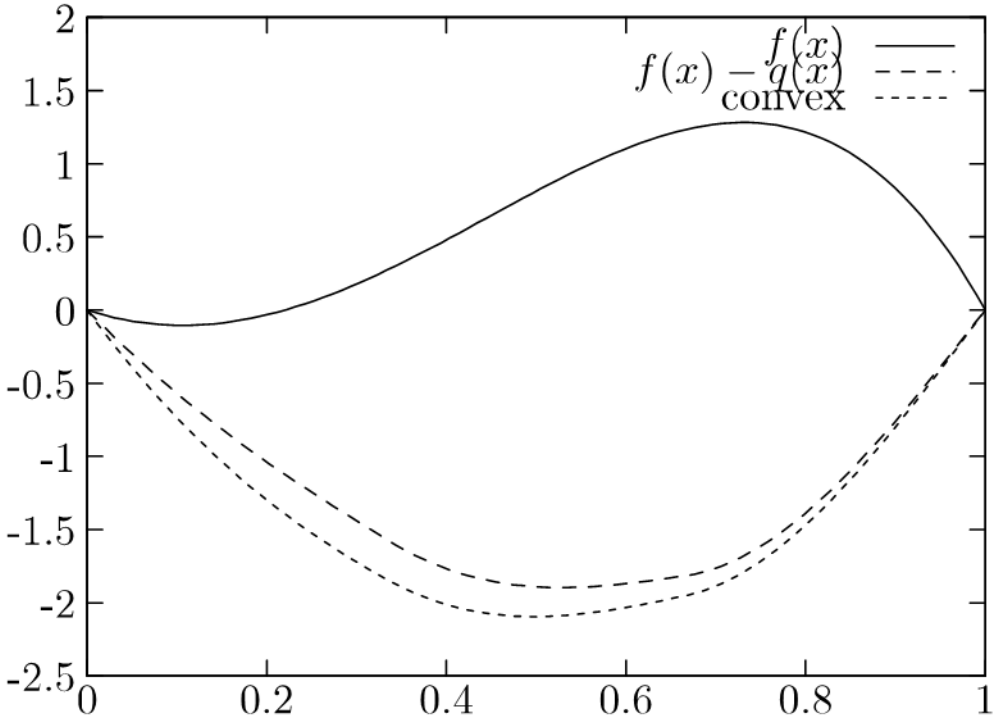

Figure 4.2: Spline α-BB underestimator.

How can we prove that $\hat{f}$ is a convex underestimator of f?
For this scope, the following two theorem are needed. The first one states that q is nonnegative over $\mathcal{R}$ (i.e., $\hat{f}$ underestimates f).

Theorem 10. *If $\alpha \geq 0$, then*

$$q(x;\alpha) \geq 0 \forall x \in \mathcal{R}$$

The second one affirms that with a suitable chosen of $\alpha_i^{k_i}$ values, $\hat{f}$ is convex.

Theorem 11. *If the $\alpha_i^{k_i}$ values, $i = 1, ..., n, k_i = 1, ..., N_i$, are large enough, then $\hat{f}$ is a convex function.*

Proof. See Locatelli and Schoen [39] for both proofs of Theorem 10 and Theorem 11. □

Another variant of the classical α-BB method is the generalized α-BB approach. Here, we will only give a brief summary; for a complete description see Akrotirianakis and Floudas [2]. In the classical α-BB approach the function q is quadratic; the new relaxation function proposed

$$q(x;\gamma) = \sum_{i=1}^{n} (1 - e^{\gamma_i(x_i - l_i)})(1 - e^{\gamma_i(u_i - x_i)})$$

gives a tighter underestimate than the quadratic function that is at least as good as that one obtained with the classical α-BB. The procedure for choosing the appropriate value of γ is based on the strict relation between the parameters γ and α; indeed it is possible shows that for some choice of γ there exists a choice of α such that $f(x) - q(x;\gamma)$ and $f(x) - q(x;\alpha)$ have the same maximum separation distance from f.

Chapter 5

Applying Domain Reduction in oBB

After giving the main notions about DR strategies, in this chapter we are going to describe how apply these approaches in oBB. Since we are solving subproblems over balls, first we give a solution based on the radius. Because of the complexity of the problems which are required to solve, a solution based on RR strategies it has been implemented. Further details about its implementation are presented in the following section.

5.1 First attempt - A Radius-based Reduction Approach

Within oBB, a global optimization problem is solved by covering the initial domain with a ball and then, at each iteration, oBB splits a ball into 3^n smaller sub-balls. Because of this, given a certain ball, the most intuitive way to reduce the domain is to attempt of reducing the radius. Given a certain node of the BB tree, i.e., given a certain ball $\mathcal{B}$ of center $x_{\mathcal{B}}$ and radius $r_{\mathcal{B}}$, before computing the lower bound one can try to reduce the radius by solving the following problem

$$\begin{aligned} \max_{x \in \mathcal{D}} & ||x - x_{\mathcal{B}}||^2 \\ & \hat{f}(x) \leq U \\ & l_{\mathcal{B}} \leq x \leq u_{\mathcal{B}} \end{aligned} \tag{5.1}$$

where $l_{\mathcal{B}}$ and $u_{\mathcal{B}}$ are the bounds of the box containing the ball and $\hat{f}(x) \leq U$ is the optimality constraint that we can remove if we consider a feasibility-based DR. Even if this is a reasonable approach for reducing the domain over a ball, we have the issue that the problem (5.1) is a global optimization problem and it has the same complexity of the original problem. Indeed,

even though there is the presence of the convex underestimator $\hat{f}$, we have to maximize a convex function (i.e. nonconvex problem); this requires the use of a global solver which is computationally too expensive.

5.2 RR in oBB

Due to the issue encountered in the radius-based reduction, a MRR strategy has been taken in consideration. As we described in Chapter 4, for what concern RR strategies, we haven't the issue related to the complexity of the problems that are required to solve. Despite this, further issues of geometric and structural nature are encountered; a more detailed description is addressed in the next session. The procedure of RR implemented in oBB is the following

Algorithm 10 : Multiple range reduction

Step 0 Let $P = \{x_1, ..., x_n\}$ be the natural of the variables and $kindRR = \{optimality, feasibility\}$. Set $l^0_{\mathcal{B}_j} = l_{\mathcal{B}_j}$, $u^0_{\mathcal{B}_j} = u_{\mathcal{B}_j}$ for $j \in \{0...n\}$ and $i = 0$.

Step 1 If $kindRR$ is *optimality*

Compute $\hat{f} =$ Spline α-BB$(f, \mathcal{D} \cap \mathcal{R}_{\mathcal{B}_i})$

Step 2 For $k = 1, ..., n$

If $kindRR$ is *optimality*, solve the two problems

$$l^{i+1}_{\mathcal{B}_k}, u^{i+1}_{\mathcal{B}_k} = \min / \max \left\{ x_k : \hat{f}_i(x) \leq U, x \in \mathcal{D} \cap \mathcal{R}_{\mathcal{B}_i} \right\},$$

Else If $kindRR$ is $feasibility$, solve the two problems

$$l^{i+1}_{\mathcal{B}_k}, u^{i+1}_{\mathcal{B}_k} = \min / \max \left\{ x_k : x \in \mathcal{D} \cap \mathcal{R}_{\mathcal{B}_i} \right\},$$

where

$$\mathcal{R}_{\mathcal{B}_i} = \left\{ x \in \mathbb{R}^n : l^i_{\mathcal{B}_k} \leq x_k \leq u^i_{\mathcal{B}_k}, l_{\mathcal{B}_j} \leq x_j \leq u_{\mathcal{B}_j}, j \in \{1...n\} \setminus \{k\} \right\},$$

is the current box (which at the beginning contain the ball $\mathcal{B}$), $\hat{f}_i$ is the underestimator of f with respect to $\mathcal{R}_{\mathcal{B}_i}$, and $l^{i+1}_{\mathcal{B}_k}, u^{i+1}_{\mathcal{B}_k}$ represent respectively the new lower and upper bounds of x_k.

Step 3 If $||l^{i+1}_{\mathcal{B}} - l^i_{\mathcal{B}}||_2 \leq tol$ and $||u^{i+1}_{\mathcal{B}} - u^i_{\mathcal{B}}||_2 \leq tol$ and $kindRR$ is *optimality* then set $i = i + 1$ and repeat Step 1. Otherwise, stop.

where both feasibility- and optimality-based RR have been implemented. For what concern the tolerance *tol* for the stopping rule, since the construction of $\hat{f}$ is rather expensive and we are not aware about the number of iteration performed by the reduction, this has been setted to $0,1$ in such a way that it is not possible produce an infinite sequence of lower and upper bounds.

Remark 7. Note that the stop criterion is considered only in the case of optimality-based RR, as well as the convex underestimator.

5.3 Adapting RR in oBB

Even though RR strategies are easily applicable and the replacement of f with his convex underestimator allows to have a pair of convex problems, adapt a box-based RR strategy inside oBB, which unlike exploits and manages balls, results in a set of issues that are needed to be handled. These problems are both of geometric structure and related to the custom implementation of oBB. First of all, there is the natural imprecision due to the fact that we start a reduction from the box containing the ball, hence, we consider a bigger region. Note that in classical oBB we have the same inconvenient for computing the lower bound since estimates the Lipschitz constant over the ball requires the bounds of the Hessian which are obtained through the bounds of the box circumscribed to the ball. In addition, another problem is related to the branching phase. In oBB, except for the first level, whenever we split nodes at the same level of the BB tree, we obtain subballs which can be in common between adjacent balls (see Figure 3.3 and 3.4). Considering a general example in Figure 5.1, oBB splits the two dashed balls $\mathcal{B}_1$ and $\mathcal{B}_2$ in subballs.

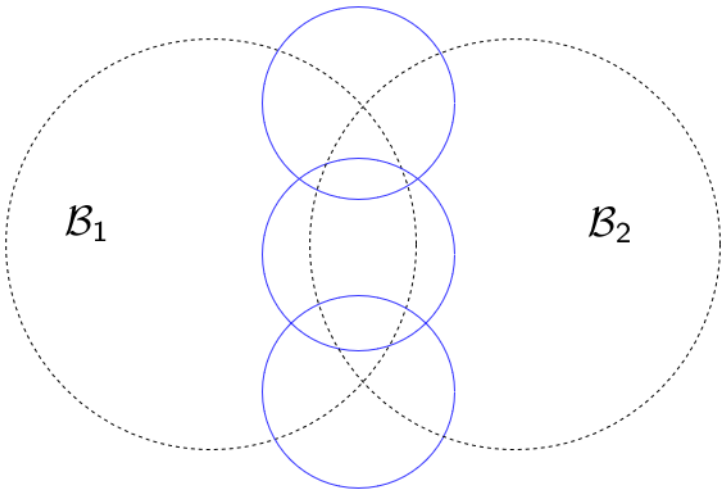

Figure 5.1: Subballs in common between two adjacent balls $\mathcal{B}_1$ and $\mathcal{B}_2$.

As we discussed in Section 3.3.2 and 3.3.3, an hash list is exploit for avoiding that several processor cores can end up bounding and splitting the same ball (the blue ones in Figure 5.1), doubling the work; in particular, by assuming that $\mathcal{B}_1$ has been split, the adjacent ball $\mathcal{B}_2$ can check the hash and avoid solving identical subproblems. With the introduction of a RR strategy, by considering a possible reduction of $\mathcal{B}_1$ and $\mathcal{B}_2$ in Figure 5.2, one can think to directly split the reduced balls. However, in this case we do not have anymore balls in common (i.e., we lose the functionality of the hash) and notwithstanding we obtain smaller subballs, the number of subproblems rapidly increases leading to a slower convergence of the algorithms.

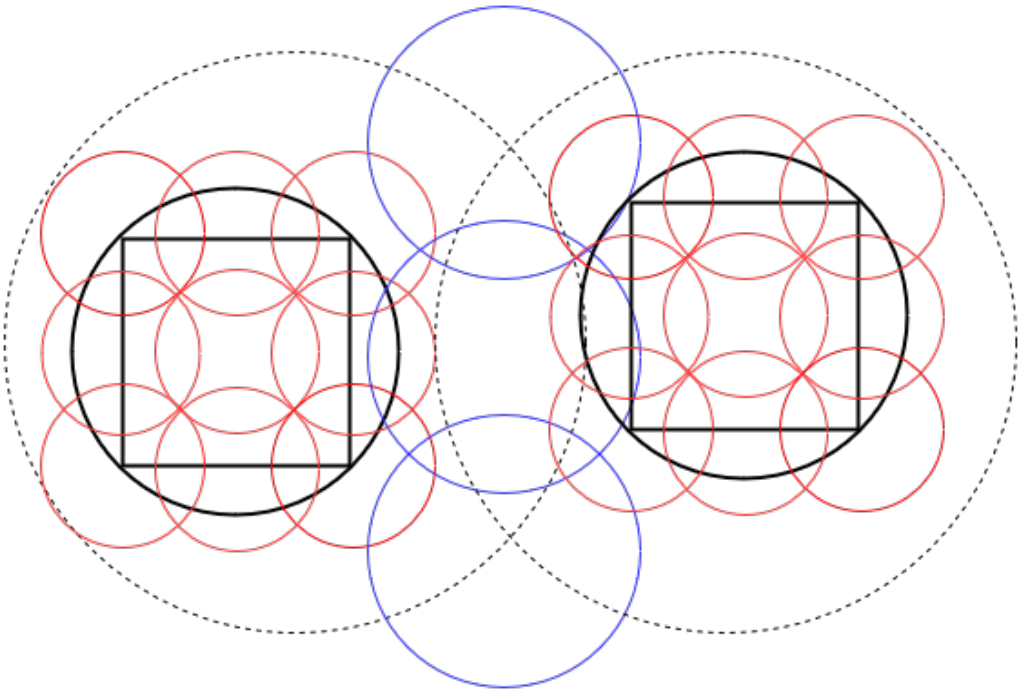

Figure 5.2: Example of RR strategy performed over the dashed balls.

The solution considered for this problem is to continue dividing the original ball maintaining a reference to the reduced box and look at the intersection when we explore the subballs. Nonetheless, when we consider the intersection, we need to consider another problem which is described in the next section.

5.4 Problem with the intersections

When we consider a certain subball and we look at the intersection, one can think to start the reduction of the subball directly from the region of intersection (see Figure 5.3).

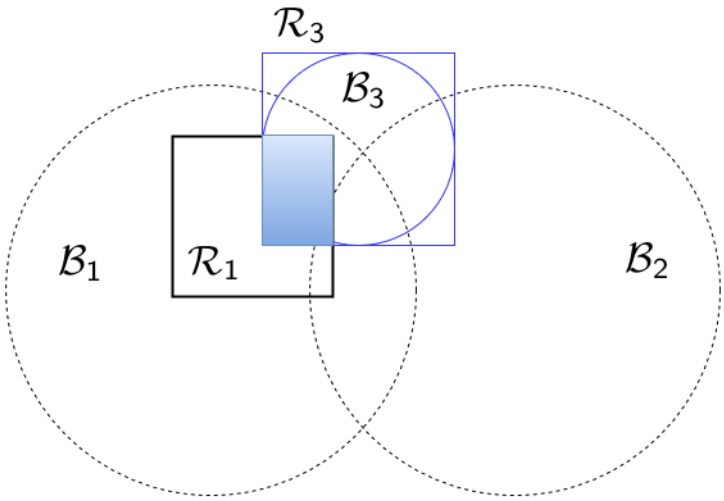

Figure 5.3: Region blue of intersection between the box $\mathcal{R}_1$ obtained by the reduction of $\mathcal{B}_1$ and the box $\mathcal{R}_3$ containing the subball $\mathcal{B}_3$

This solution, that might seem the most appropriate, is possible at the first step of splitting (because there aren't overlapping balls) but, in the other steps, leads to a loss of regions within the domain that should be instead considered. Indeed, by assuming, for example, that $\mathcal{B}_3$ has been created starting from $\mathcal{B}_1$, if we take care only about the intersection between $\mathcal{R}_1$ and $\mathcal{R}_3$, we lose the intersection between $\mathcal{R}_2$ and $\mathcal{R}_3$ (see Figure 5.4).

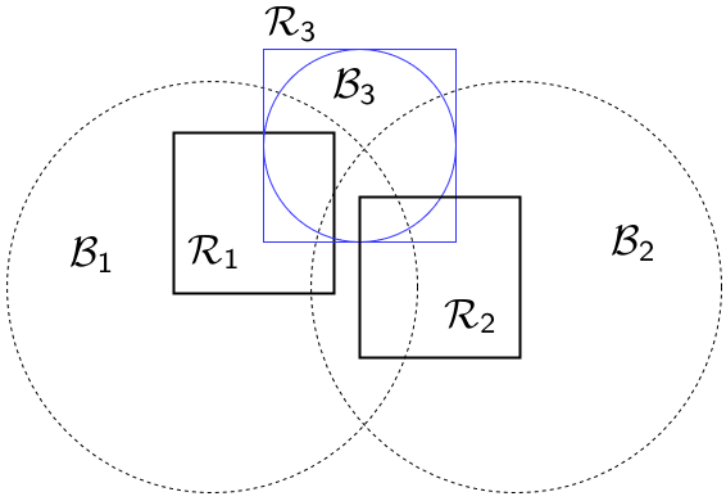

Figure 5.4: Graphical representation of the intersection problem.

Our solution is to check the intersection between the subball and the reduced domain of the ball from which the subball has been created and, eventually,

take the part of subball entirely contained in the original domain $\mathcal{D}$ (because is useless consider the part outside which will never contain the global minimum) for his reduction and the following computation of the lower bound. Formally, given the example in Figure 5.4 and by assuming, without loss of generality, that we start the splitting from $\mathcal{B}_1$, we have the following situation

- if $\mathcal{R}_3 \cap \mathcal{R}_1 \neq \emptyset$: consider $\mathcal{R}_3 \cap \mathcal{R}$ and apply the RR strategy, where $\mathcal{R}$ is the box of the original problem;
- otherwise: discard the subball (i.e., do not insert in the hash list) in such a way that $\mathcal{B}_2$ can create $\mathcal{B}_3$ (from the splitting step) and check $\mathcal{R}_2 \cap \mathcal{R}_3$. In this case, if $\mathcal{R}_2 \cap \mathcal{R}_3 \neq \emptyset$, consider $\mathcal{R}_3 \cap \mathcal{R}$ and apply the RR strategy; else, discard the subball.

In this way, even if, we do not fully exploit the power of the reduction, we are completely sure of not discarding regions that might contain the global optimum. A possible improvement to eliminate this inaccuracy would require that each subball is always created starting from a single ball (i.e., absence of overlapping balls); this would imply of starting the subdivision from the reduced domain, leading to the issue of the high number of subproblems previously described in Section 5.3.

5.5 Comparing oBB with and without RR: a simple example

In this section we are going to compare the results between classical oBB and the modified version with the RR strategy embedded inside. In particular, since we are focused to study the effects produced by RR, results in terms of lower bound values are considered. For this scope we have considered the following global optimization problem (sum of sines):

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \sum_{i=1}^{n} \sin x_i \\ -1 \leq x_i \leq 1 \; \forall i = 1, ..., n \\ \sum_{i=1}^{n} -x_i \leq 1 \end{aligned} \tag{5.2}$$

with $n = 2$. Table 5.1 shows the value of the minimum lower bound produced at each step of the algorithm with one worker processor by comparing the results produced by classical oBB, oBB with the execution of only optimality-based RR at each level of the BB tree and oBB with the execution of only feasibility-based RR at each level of the BB tree.

Table 5.1: Lower bounds values produced at each step by the original oBB, oBB with optimality-based RR and oBB with feasibility-based RR.

Standard	Feas-based	Opt-based
-5.703901	-13.166259	-13.166259
-5.373720	-5.387231	-1.982494
-5.154266	-4.793479	-1.956202
-4.051223	-4.779087	-1.580084
-3.529755	-4.481481	-1.207985
-3.344375	-2.128153	-1.142538
-3.131878	-2.126185	-1.056478
-1.765673	-2.037619	-1.009295
-1.645759	-1.700354	-0.995364
-1.560090	-1.696013	-1.001399
-1.538259	-1.575959	-0.994645
-1.289582	-1.490197	-0.984564
-1.282386	-1.441866	-0.976171
-1.206977	-1.441866	-0.970613
-1.188957	-1.441866	-0.969826
-1.150753	-1.234429	-0.979499
-1.149942	-1.211917	-0.969734
-1.149753	-1.169771	-0.969110
-1.112929	-1.152521	
-1.083164	-1.144208	
-1.066012	-1.144208	
-1.063118	-1.132707	
-1.050250	-1.116680	
-1.044158	-1.096609	
-1.028152	-1.096609	
-1.020112	-1956929	
-1.019877	-1.054581	
-1.011050	-1.052581	
-1.010585	-1.046665	
-1.005868	-1.046254	
-1.004900	-1.030097	
-0.996350	-1.022300	
-0.994666	-1.012808	
-0.991242	-1.011342	
-0.986383	-1.010802	
-0.984564	-1.006223	
-0.984211	-1.005176	

Table 5.1: continue in the next page

Table 5.1: continue from the previous page

Standard	Feas-based	Opt-based
-0.983979	-1.004577	
-0.983699	-0.997694	
-0.981938	-0.996620	
-0.981455	-0.994922	
-0.980861	-0.990960	
-0.978586	-0.986682	
-0.977946	-0.984589	
-0.977598	-0.984008	
-0.973857	-0.983726	
-0.973295	-0.981982	
-0.971113	-0.981487	
-0.970988	-0.981196	
-0.970868	-0.981137	
-0.970643	-0.981068	
-0.970492	-0.979334	
-0.970262	-0.978516	
-0.969623	-0.977988	
-0.969307	-0.977869	
-0.968723	-0.977763	
-0.968388	-0.977705	
-0.968017	-0.973917	
-0.967630	-0.973348	
	-0.972401	
	-0.971116	
	-0.970992	
	-0.970871	
	-0.970651	
	-0.970496	
	-0.970266	
	-0.969820	
	-0.969629	
	-0.969312	
	-0.968800	
	-0.968780	
	-0.968778	
	-0.968657	
	-0.968397	
	-0.968024	
	-0.967689	

One can see from Table 5.1 as in most cases the value of the lower bound obtained with the application of the feasibility-based RR is worse, i.e., the lower bound is lower. Indeed, since the value of global upper bound should be the same amongst the three version of the algorithm, the Optimality Gap $\gamma = U - L$, in most cases is higher. The same conclusion can be reached if we consider that the number of steps performed by oBB with feasibility-based RR is greater, hence, the convergence of this version is slower since it takes longer time to satisfy the stopping rule. Such unexpected behavior is due, most likely, to the poor compatibility between boxes and balls; in fact, as shown in Figure 5.5 very often it happens that the ball around the box obtained by the reduction is not completely inside the original box causing an error in the computation of the lower bound.

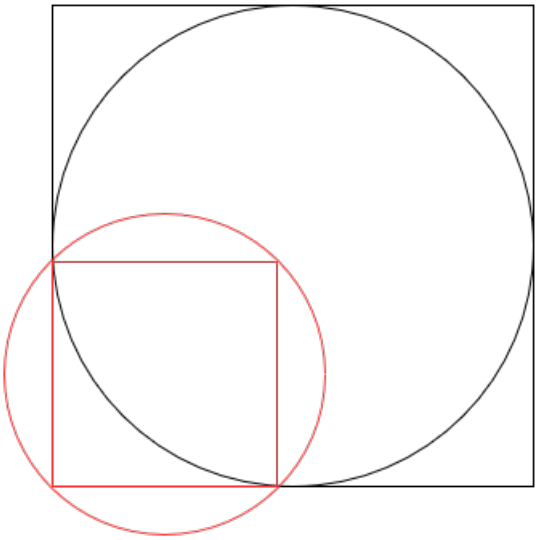

Figure 5.5: Example of reduction which produces a ball partially outside the original box. Due to the poor affinity between boxes and balls, every time is performed a reduction in the direction of an edge, the ball around the resulting domain might fall outside the original box. This behavior has the effect of computing a lower bound that can be lower or at most equal to that one computed by considering a ball completely inside the original ball.

Such issue, that can be difficult to manage, can be partially solved if we consider a smaller ball. In particular, if the ball around the reduced box is not completely inside the original box, we consider the inside ball as shown in Figure 5.6. As just said this solution partially solves the problem but some inaccuracy remains, indeed:

- even if completely inside the original box, the ball might fall outside the original ball (possible optimality gap higher);

- the ball does not completely cover the reduced box near the corners (possible optimality gap lower).

Particularly, as we will see in Section 5.6, if the latter case happens very often at the first levels of the BB tree, the algorithm may converge too early, producing an inaccurate solution; i.e., since the lower bounds are higher, fewer steps are required to satisfy the stopping rule and arrest the algorithm.

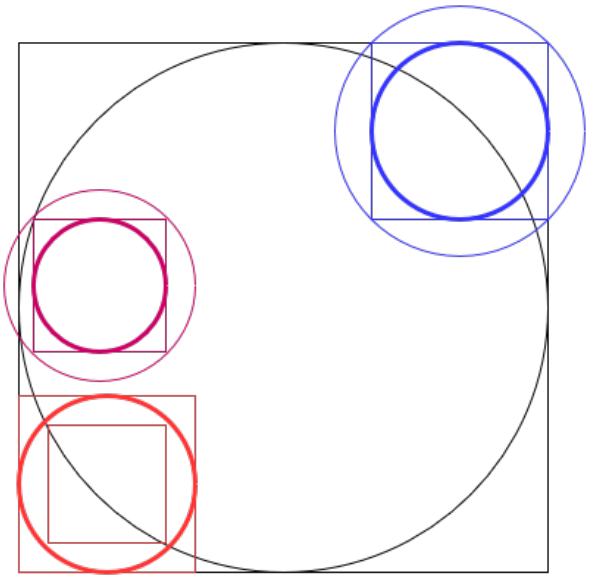

Figure 5.6: Some examples of possible reductions which produce a ball partially outside of the original box. In this case an inside ball it is considered.

In relation to the results of Table 5.1, Table 5.2 shows the modified results by considering the proposed solution. Such solution allows to have better lower bounds (in terms of optimality gap) for both RRs at each step; in particular optimality-based RR greatly improves the lower bound in the first few iterations. As a consequence, even feasibility-based RR reaches the termination in a less number of steps respect to canonical oBB. Furthermore, through the checking of the intersection, both the RR strategies permit of discarding more balls than Standard Obb as represented in Figure 5.7. Notwithstanding, both the form of reduction produce more accurate lower bounds and decrease the number of subballs, another important factor to consider when we use the reduction strategies is the computational time. As RR procedures are of varying complexity, those ones that consume most of the time are not performed at all nodes but only at the root node or up to a limited depth. Faster but weaker RR procedures can instead be performed at all nodes. As already described in the previous chapter, the optimality-based RR procedure identifies the smallest range of each variables

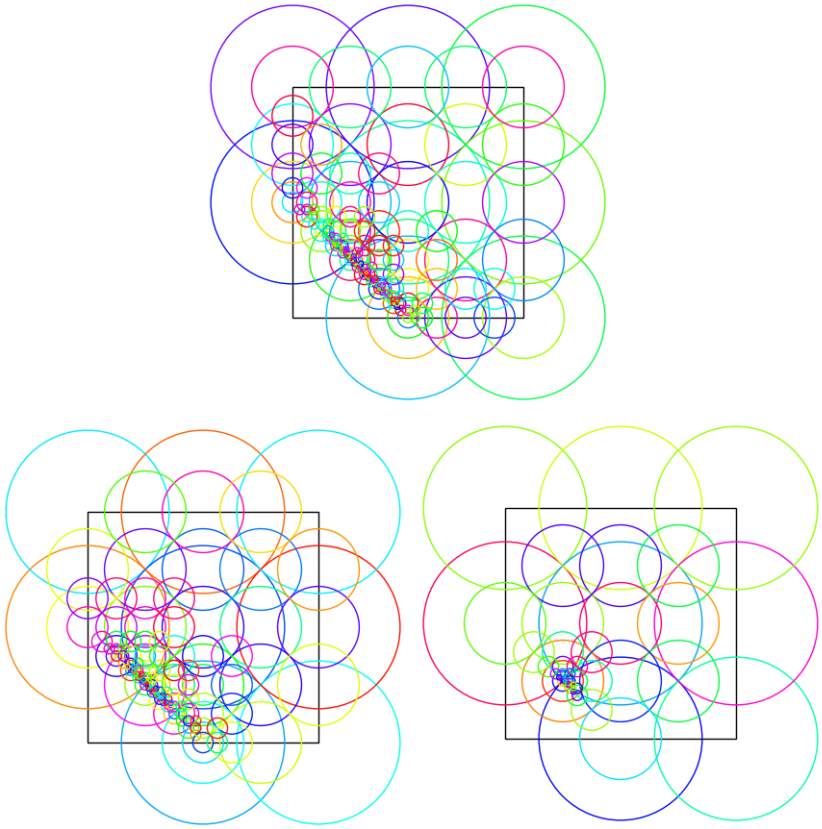

Figure 5.7: Graphical Results of oBB (up), oBB with feasibility-based RR (bottom left) and oBB with optimality-based RR (bottom right).

ensuring that the algorithm will not have to explore regions which do not contain any feasible point. Unfortunately, as shown in Table 5.2, this is a computationally expensive procedure and because of the associated cost, we have decided to apply this type of reduction only at the first two levels of depth. The feasibility-based RR procedure which is computationally cheaper than the optimality-based one, is performed instead through the other levels of the BB tree. Further results obtained by using this version of reduction are shown and described in the next section. Another possible solution implemented in COUENNE (see Belotti et al. [6]), is to use optimality-based RR in all nodes of the BB tree up to a certain depth, specified by a parameter L_{abt} (that is typically 2 in COUENNE), and with probability $2^{L_{abt}-\lambda}$ for nodes at depth $\lambda > L_{abt}$.

Table 5.2: Lower bounds values produced at each step by the original oBB, oBB with optimality-based RR and oBB with feasibility-based RR by considering the proposed solution of taking the inside ball.

Standard	Feas-based	Opt-based
-5.703901	-5.154266	-5.154266
-5.373720	-2.307532	-1.812918
-5.154266	-2.225189	-1.412717
-4.051223	-1.879738	-1.297691
-3.529755	-1.672939	-1.117798
-3.344375	-1.463470	-1.105788
-3.131878	-1.289582	-1.056478
-1.765673	-1.282386	-1.009295
-1.645759	-1.117678	-0.995364
-1.560090	-1.104916	-0.994645
-1.538259	-1.096220	-0.993536
-1.289582	-1.065587	-0.976609
-1.282386	-1.056659	-0.969734
-1.206977	-1.027580	-0.968399
-1.188957	-1.024063	
-1.150753	-1.011896	
-1.149942	-1.004231	
-1.149753	-0.992040	
-1.112929	-0.991674	
-1.083164	-0.991242	
-1.066012	-0.988165	
-1.063118	-0.986655	
-1.050250	-0.985979	
-1.044158	-0.981807	
-1.028152	-0.978586	
-1.020112	-0.976954	
-1.019877	-0.975705	
-1.011050	-0.975067	
-1.010585	-0.974460	
-1.005868	-0.974215	
-1.004900	-0.972383	
-0.996350	-0.971975	
-0.994666	-0.969325	
-0.991242	-0.968871	
-0.986383	-0.968467	
-0.984564		

Table 5.2: continue in the next page

Table 5.2: continue from the previous page

Standard	Feas-based	Opt-based
-0.984211		
-0.983979		
-0.983699		
-0.981938		
-0.981455		
-0.980861		
-0.978586		
-0.977946		
-0.977598		
-0.973857		
-0.973295		
-0.971113		
-0.970988		
-0.970868		
-0.970643		
-0.970492		
-0.970262		
-0.969623		
-0.969307		
-0.968723		
-0.968388		
-0.968017		
-0.967630		
T: 4.958s	T: 6.321s	T:313.20s

5.6 Numerical Results: COCONUT Test Set

For a more thorough numerical evaluation, the algorithm has been run on a set of problems from the COCONUT benchmark, considering the implementation of reduction defined at the end of the previous section (i.e., optimality-based RR for the first two levels and then feasibility-based RR). Throughout the rest of the book we will call Hybrid Range Reduction (HRR) this kind of reduction strategy. The hardware used is part of the HPC Arcus Phase B cluster consisting of Dual Haswell CPU nodes which have 16 cores per node and a minimum of 64Gb of memory. Table 5.3 summarizes the execution time and the solution $f(x^*)$ obtained by oBB and oBB with HRR. Looking at the performance of both version, we can see how, in most cases, the application of HRR significantly decreases the execution times of the algorithm with an average speedup of around 4.6 times.

Table 5.3: Execution time and solution obtained running oBB and oBB with HRR on a RBF approximations to selected problems from the COCONUT test set. All problems have been run to the absolute tolerance they achieved in 12 hours on the serial code.
The authors would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this work. `http://dx.doi.org/10.5281/zenodo.22558`

	Obb		Obb + HRR	
Problem	Ubound	Time (s)	Ubound	Time (s)
biggs5	-11515.618661	1459.76	-11418.464208	689.405
biggsc4	-24.596506	896.015	-24.596506	620.255
brownden	-67152761.858340	734.884	-67152761.978252	627.494
bt3	3.203325	1258.63	3.203325	1299.56
ex2_1_1	-11.132269	1140.94	-11.130924	354.288
ex2_1_2	-212.877397	1274.66	-209.754114	19.877
ex2_1_4	-10.999721	2176.11	-10.999476	221.792
ex6_2_10	-2.980167	1416.44	-2.973069	685.187
ex6_2_13	-0.233242	1913.04	-0.233242	1440.99
expfita	-895.265254	1328.92	-797.105479	194.370
expfitb	-116545.892857	1302.06	-88490.406105	941.737
expfitc	-875848.280936	1472.44	-889715.667899	399.954
hatflda	0.143638	577.426	0.143638	631.485
hatfldb	-0.227838	431.168	0.066854	198.100
hatfldc	-218.544212	1016.46	-209.575434	727.375
hatfldh	-24.596506	884.501	-24.596506	611.684
hong	-6.101767	816.924	-5.688681	554.082
hs038	-68735.300540	812.261	-67922.998061	779.403
hs041	1.927569	242.809	1.927569	146.930
hs045	1.492050	1092.37	1.543665	150.938
hs048	1.014754	1485.45	1.014754	1519.08
hs049	-163.228099	1685.48	-160.821383	121.108
hs051	0.466116	971.230	0.466116	951.465
hs052	12.564294	1395.83	12.564294	1362.17
hs053	0.309577	1241.49	0.309577	1160.78
hs054	-0.377621	1494.17	-0.377621	534.418
hs055	6.340291	447.384	6.340291	444.066
hs086	2.365612	1415.35	2.365612	1825.67
hs268	-5311.648213	1637.23	-4394.230416	991.523
kowosb	-24664.590507	960.860	-24664.590507	766.109
lsnnodoc	113.273339	321.960	113.273339	239.332

For what concern those few cases for which the computational time is not improved (e.g., bt3, hatflda or hs086), this is due to the fact that, sometimes, HRR might slightly reduces or not cut at all the feasible set; in this case we lose the compromise between the benefits led and the effort of its application increasing the computational time. Notwithstanding the improvements achieved with HRR, in several cases (highlighted in red in Table 5.3), the solution obtained is not as accurate as the one identified by classical oBB. This is very disappointing but not unexpected as bounding subproblems can arise the issues due to imprecision on the computation of lower bounds (see Figure 5.6 and its relative explanation).

Remark 8. Note that, in most of the example for which the upper bound is different, the distance of the obtained RBF solution with classical oBB to the best known solution for the original problem is greater than 1 (See Table A.5 in Cartis et al. [10] for more details).

For avoiding the problem pointed out in Table 5.3 several ideas have been taken in consideration. The following two that we are going to describe are based on the fact that by limiting the issue at the first levels of the BB tree, where the subregions are larger, a minor error in the computation of the lower bound should be committed. The first considers a smaller absolute tolerance for the stopping rule; particularly, let ϵ be the fixed absolute tolerance in classical oBB, we set $\epsilon = \epsilon - 20\%\epsilon$ for our variant. In this way, even if the algorithm computes higher lower bounds, it is less likely that balls are discarded quickly. Unfortunately, the main disadvantage of this idea is that, even the balls which do not contain the optimal solution are explored in depth increasing drastically the execution times. The latter idea directly exploits the geometrical issue encountered. Indeed, such problem can be solved if we revert to the original ball, instead of using a ball inside the reduced box. Table 5.4 shows the results of oBB with the application of this enhancement up to a fixed level of the BB tree. In particular, it is possible to note how increasing the number of levels lessens the probability of computing an inaccurate lower bound and, thus, increases the probability of reaching a better upper bound; indeed, one can see how the number of match with original oBB is higher by setting the depth to 3. As expected, with the increment of the number of levels, also the computational time increases; in fact, by reverting to the original ball, it is ignored the effect brought by the reduction. In particular, by taking in exam only the problems of Table 5.4, oBB+HRR is on average 3.2 times faster than oBB+HRR(3), while oBB+HRR(2) preserves on average the same speedup of oBB+HRR. Obviously, such idea can be extended for the entire depth of the BB tree; this would allow to reach the same results of classical oBB for all the test set obtaining, however, a marked deterioration of the times.

In conclusion, we can clearly see from the numerical results of Table 5.3 that a RR approach leads to an efficient parallel algorithm on average which exhibits good speedup. Due to a difficult adjustment between boxes and balls, several issues remain, especially the inaccuracy on the computation of lower bounds that does not allow to always reach the same solution of classical oBB. This is due to the fact that by taking an inside ball (when that one circumscribed the reduced box is not completely inside the orginal box) the regions in proximity of the corners of the reduced box are not considered for the computation of the lower bound. Nonetheless, it is possible remove this problem (by reverting to the original ball) obtaining, however, an increase of the execution time. Overall, we have shown that it is possible to design an efficient parallel overlapping branch and bound algorithm with a range reduction strategy even though after overcoming some underlying difficulties.

Table 5.4: Execution time and solution obtained running oBB and oBB with HRR on a RBF approximations to selected problems from the COCONUT test set. The numbers in brackets represent the number of levels of the BB tree for which it is used the idea of reverting to the original ball. All problems have been run to the absolute tolerance they achieved in 12 hours on the serial code. **The authors would like to acknowledge the use of the University of Oxford Advanced Research Computing (ARC) facility in carrying out this work.** `http://dx.doi.org/10.5281/zenodo.22558`

	Obb		Obb + HRR (2)		Obb + HRR (3)	
Problem	Ubound	Time (s)	Ubound	Time (s)	Ubound	Time (s)
biggs5	-11515.618661	1459.76	-11418.464208	638.476	-11515.618661	2005.19
ex2_1_1	-11.132269	1140.94	-11.132269	347.092	-11.132269	1405.11
ex2_1_2	-212.877397	1274.66	-209.754114	19.128	-209.754114	17.900
ex2_1_4	-10.999721	2176.11	-10.999476	297.188	-10.999476	2066.84
ex6_2_10	-2.980167	1416.44	-2.973069	728.966	-2.980167	1692.50
expfita	-895.265254	1328.92	-797.105479	191.534	-798.864509	984.859
expfitb	-116545.892857	1302.06	-88490.406105	803.518	-89141.528976	1962.66
hatfldb	-0.227838	431.168	0.066854	195.098	0.066854	253.085
hatfldc	-218.544212	1016.46	-209.575434	664.425	-218.544212	1214.14
hong	-6.101767	816.924	-5.688681	528.919	-5.688681	548.489
hs038	-68735.300540	812.261	-67922.998061	746.850	-68735.300540	977.355
hs045	1.492050	1092.37	1.492050	213.248	1.492050	1198.49
hs049	-163.228099	1685.48	-160.821383	131.547	-163.228108	370.123
hs268	-5311.648213	1637.23	-4394.230416	1006.85	-5311.649095	1834.73

Conclusions

We have presented improvements of a branch and bound algorithm presented in Fowkes et al. [25] for the minimisation of a twice differentiable nonconvex objective function with a Lipschitz continuous Hessian over a compact, convex set. Cartis et al. [10] have shown that it is possible to improve the bounding strategies by exploiting a variety of both existing and novel bounds and the branching procedure through the use of parallelism. Their results prove that no single bound is optimal, albeit in general the second order lower bound exhibits the best performance. For what concern instead the branching phase two standard paradigms, namely data parallelism and task parallelism, have been studied. The first, which parallelize the bounding operations within the algorithm, performed poorly, while the latter, which parallelize the branch and bound tree, exhibited excellent average speedup on a large number of test problems.

Our work, has shown that it is possible to obtain further improvements on the computational times through the application of a domain reduction strategy. Such strategy, used for cutting the feasible region without cutting the global optimum solution, allow to have sharper underestimator which has the consequence of improving the lower bounds. Because of the nonconvex problem that need to be solved in a reduction based on the radius of the ball, a boxes-based hybrid range reduction strategy has been implemented. Several issues, related to the poor compatibility between boxes and balls have been addressed. The most important regards the fact that wrong lower bounds can be computed if the reduced box is positioned in such a way that its circumscribed ball gets out the original box. The idea followed for this scope, is to take a ball inside the reduced box, even though, it partially solves the problem because by taking an inside ball the regions close to the corners of the box are not covered and, thus, not considered for the computation of the lower bound. Such imprecision may lead the algorithm to converge too early, producing an inaccurate solution. A possible fix could be that one of reverting to the original ball instead of taking an inside ball, but in this case the improvements on the execution times are lost. Numerical results have shown an appreciable improvement in terms of computational time, however, the original method seem to be still superior

in terms of quality of the solution obtained.

Since the approach is limited to low dimensional problems and problems where Hessian Lipschitz constant is available or can be approximated, a critical challenge remains: scaling up the problem dimension in such a way that problems of higher practical interest can be solved. A good remedy perhaps lies in considering different criteria for covering domain avoiding overlapping but still allowing to use nonconvex lower bounds. However, further improvements can be considered such as use a cheaper way to compute the convex underestimator for the range reduction strategy, consider different domain reduction strategies or try to make tractable the global optimization problem within the radius-based reduction.

Appendix A

Notation

Given a twice continuously differentiable function $f : \mathcal{C} \to \mathbb{R}$, where $\mathcal{C} \subset \mathbb{R}^n$ is a sufficiently large open set containing the convex, compact domain $\mathcal{D}$; let $g(x) := \nabla_x f(x)$ and $H(X) := \nabla_{xx} f(x)$ be the gradient and Hessian of $f(x)$, and $\|\cdot\|$ denote the l_2-norm. By considering that C^k is the space of k-continuously differentiable functions; since $\mathcal{C}$ is compact and $f \in C^2(\mathcal{C})$, there are constants $L_f > 0$ and $L_g > 0$ for which

$$\|g(x)\| \leq L_f \text{ and } \|H(x)\| \leq L_g \tag{A.1}$$

for all $x \in \mathcal{C}$. It follows that L_f and L_g are respectively the l_2-norm Lipschitz constant and the gradient Lipschitz constant for $f(x)$, and thus

$$|f(x) - f(y)| \leq L_f \|x - y\| \text{ and } \|g(x) - g(y)\| \leq L_g \|x - y\| \tag{A.2}$$

for all $x, y \in \mathcal{C}$. In addition, since we have assumed that f has a Lipschitz continuous Hessian, we can define a Hessian Lipschitz constant $L_H > 0$ for f over $\mathcal{C}$ such that

$$\|H(x) - H(y)\| \leq L_H \|x - y\|$$

for all $x, y \in \mathcal{C}$, where the $\|\cdot\|$ on the left hand side denotes the induced matrix norm. Note that, within $\mathcal{C}$, these are all global Lipschitz constants. Furthermore, by denoting by $\mathcal{B} \subset \mathcal{C}$ the n-dimensional closed ball of radius $r_{\mathcal{B}} > 0$ centred at some $x_{\mathcal{B}} \in \mathcal{B}$, i.e.

$$\mathcal{B} = \{x \in \mathbb{R}^n : \|x - x_{\mathcal{B}}\| \leq r_{\mathcal{B}}\},$$

we can define a local Gradient $L_g(\mathcal{B})$ and Hessian Lipschitz constant $L_H(\mathcal{B})$ for $f(x)$ over the ball $\mathcal{B}$ whose, unlike the global Lipschitz constants, we need to be able to calculate a numerical approximation for the proposed algorithm.

Appendix B

First order lower bound

Given the following assumption about problem (3.1):

Assumption 2. The objective function $f : \mathcal{C} \to \mathbb{R}$ is twice continuously differentiable, where $\mathcal{C} \subset \mathbb{R}^n$ is a sufficiently large open set containing the convex, compact domain $\mathcal{D}$.

the following result shows why lower bounds on $\lambda_{min}(H(x))$ can be used in place of the gradient Lipschitz constant $-L_g$ in (3.2).

Lemma 1 (Evtushenko and Posypkin, 2013 [19])**.** *Let AF 2 hold. Suppose $\mathcal{B} \subset \mathcal{C}$ is a convex, compact subdomain and $x_{\mathcal{B}} \in \mathcal{B}$. Then, for any $x \in \mathcal{B}$ we have*

$$f(x) \geq f(x_{\mathcal{B}}) + (x - x_{\mathcal{B}})^T g(x_{\mathcal{B}}) + \frac{\lambda^{\mathcal{B}}_{min}(H)}{2}\|x - x_{\mathcal{B}}\|_2^2 \tag{B.1}$$

where

$$\lambda^{\mathcal{B}}_{min}(H) := \min_{\xi \in \mathcal{B}} \lambda_{min}(H(\xi)). \tag{B.2}$$

Proof. For all $x, x_{\mathcal{B}} \in \mathcal{B}$ and some $\xi(x) \in \mathcal{B}$ the first order Taylor expansion with the Lagrange form for the remainder gives

$$\begin{aligned}
f(x) &= f(x_{\mathcal{B}}) + (x - x_{\mathcal{B}})^T g(x_{\mathcal{B}}) + \frac{1}{2}(x - x_{\mathcal{B}})^T H(\xi)(x - x_{\mathcal{B}}) \\
&= f(x_{\mathcal{B}}) + (x - x_{\mathcal{B}})^T g(x_{\mathcal{B}}) + \frac{1}{2}\frac{(x - x_{\mathcal{B}})^T H(\xi)(x - x_{\mathcal{B}})}{(x - x_{\mathcal{B}})^T (x - x_{\mathcal{B}})}(x - x_{\mathcal{B}})^T (x - x_{\mathcal{B}}) \\
&\geq f(x_{\mathcal{B}}) + (x - x_{\mathcal{B}})^T g(x_{\mathcal{B}}) + \frac{\lambda_{min}(H(\xi))}{2}\|x - x_{\mathcal{B}}\|_2^2 \\
&\geq f(x_{\mathcal{B}}) + (x - x_{\mathcal{B}})^T g(x_{\mathcal{B}}) + \frac{\lambda^{\mathcal{B}}_{min}(H)}{2}\|x - x_{\mathcal{B}}\|_2^2
\end{aligned}$$

where the last two inequalities follow from the fact that the Rayleigh quotient reaches its minimum at the smallest eigenvalue and from (B.2), respectively. □

Appendix C

Second order lower bound

Let $T \in \mathbb{R}^{n\times n\times n}$ denote a third order tensor (i.e. 3-dimensional array) and let t_{ijk} be the (i,j,k)-th component (i.e. element in the array) of the tensor T. A tensor T is called symmetric if $t_{\sigma(i)\sigma(j)\sigma(k)} = t_{ijk}$ for any permutation σ of the indices (i,j,k). This is the natural generalisation of a symmetric matrix to tensors. For a vector $x \in \mathbb{R}^n$, the multiplication of a tensor T three times on the right by x is denoted by

$$Tx^3 := \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} t_{ijk}x_ix_jx_k.$$

Let $\|T\|_F$ denote the Frobenius norm for the tensor T defined as

$$\|T\|_F^2 = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} t_{ijk2}^2.$$

We have from Lim, 2005 [37] that the multilinear Rayleigh quotient for the l^3-norm is given by

$$\frac{Tx^3}{\|x\|_3^3},$$

where $\|\cdot\|_3$ is the l^3-norm for vectors. Furthermore, the l^3-eigenvalues (or H-eigenvalues) of T are defined as the stationary points of the multilinear Rayleigh quotient, that is, the smallest l^3-eigenvalue of T, $\lambda_{min}^{l^3}(T)$ is given by

$$\lambda_{min}^{l^3}(T) = \min_{x\neq 0} \frac{Tx^3}{\|x\|_3^3}. \tag{C.1}$$

By assuming the following about problem (3.1):

Assumption 3. The objective function $f : \mathcal{C} \to R$ is thrice continuously differentiable, where $\mathcal{C} \subset \mathbb{R}^n$ is a sufficiently large open set containing the convex, compact domain $\mathcal{D}$.

and by denoting

$$T(x) := \nabla_{xxx} f(x)$$

as the third order derivative tensor of $f(x)$ (which is symmetric by construction), we can show why lower bounds on the spectrum of the derivative tensor can be used in place of the Hessian Lipschitz constant L_H in (3.3).

Lemma 2. *Let AF 3 hold. Suppose $\mathcal{B} \subset \mathcal{C}$ is a convex, compact subdomain and $x_{\mathcal{B}} \in \mathcal{B}$. Then, for any $x \in \mathcal{B}$ we have*

$$\begin{aligned} f(x) \geq f(x_{\mathcal{B}}) + (x - x_{\mathcal{B}})^T g(x_{\mathcal{B}}) + \frac{1}{2}(x - x_{\mathcal{B}})^T H(x_{\mathcal{B}})(x - x_{\mathcal{B}}) \\ + \begin{cases} \frac{\lambda_{min}^{l^3,\mathcal{B}}(T)}{6} \|x - x_{\mathcal{B}}\|_2^3 & \text{if } \lambda_{min}^{l^3,\mathcal{B}}(T) \leq 0 \\ \frac{\lambda_{min}^{l^3,\mathcal{B}}(T)}{6} n^{-1/2} \|x - x_{\mathcal{B}}\|_2^3 & \text{if } \lambda_{min}^{l^3,\mathcal{B}}(T) > 0 \end{cases} \end{aligned} \tag{C.2}$$

where

$$\lambda_{min}^{l^3,\mathcal{B}}(T) = \min_{\xi \in \mathcal{B}} \lambda_{min}^{l^3}(T(\xi)). \tag{C.3}$$

Proof. The proof requires relations between the l^2 and l^3 vector norms. It is a standard result that for any $p > r > 0$

$$\|x\|_p \leq \|x\|_p \leq n^{(1/r - 1/p)} \|x\|_p$$

for any $x \in \mathbb{R}^n$. This means that

$$\begin{aligned} \|x\|_3 \geq n^{-1/6} \|x\|_2, \\ \|x\|_3 \leq \|x\|_p \end{aligned} \tag{C.4}$$

for any $x \in \mathbb{R}^n$.
For $x = x_{\mathcal{B}}$ the claim in the theorem is trivial, so w.l.o.g. assume $x \neq x_{\mathcal{B}}$. Then for all $x, x_{\mathcal{B}} \in \mathcal{B}$ and some $\xi(x) \in \mathcal{B}$, the second order Taylor expansion with the Lagrange form for the remainder gives

$$\begin{aligned} f(x) &= f(x_{\mathcal{B}}) + (x - x_{\mathcal{B}})^T g(x_{\mathcal{B}}) + \frac{1}{2}(x - x_{\mathcal{B}})^T H(x_{\mathcal{B}})(x - x_{\mathcal{B}}) + \frac{1}{6} T(\xi)(x - x_{\mathcal{B}})^3 \\ &= f(x_{\mathcal{B}}) + (x - x_{\mathcal{B}})^T g(x_{\mathcal{B}}) + \frac{1}{2}(x - x_{\mathcal{B}})^T H(x_{\mathcal{B}})(x - x_{\mathcal{B}}) + \frac{1}{6} \frac{T(\xi)(x - x_{\mathcal{B}})^3}{\|x - x_{\mathcal{B}}\|_3^3} \|x - x_{\mathcal{B}}\|_3^3 \\ &\geq f(x_{\mathcal{B}}) + (x - x_{\mathcal{B}})^T g(x_{\mathcal{B}}) + \frac{1}{2}(x - x_{\mathcal{B}})^T H(x_{\mathcal{B}})(x - x_{\mathcal{B}}) + \frac{\lambda_{min}^{l^3}(T(\xi))}{6} \|x - x_{\mathcal{B}}\|_3^3 \\ &\geq f(x_{\mathcal{B}}) + (x - x_{\mathcal{B}})^T g(x_{\mathcal{B}}) + \frac{1}{2}(x - x_{\mathcal{B}})^T H(x_{\mathcal{B}})(x - x_{\mathcal{B}}) + \frac{\lambda_{min}^{l^3,\mathcal{B}}(T)}{6} \|x - x_{\mathcal{B}}\|_3^3 \\ &\geq f(x_{\mathcal{B}}) + (x - x_{\mathcal{B}})^T g(x_{\mathcal{B}}) + \frac{1}{2}(x - x_{\mathcal{B}})^T H(x_{\mathcal{B}})(x - x_{\mathcal{B}}) \\ &\quad + \begin{cases} \frac{\lambda_{min}^{l^3,\mathcal{B}}(T)}{6} \|x - x_{\mathcal{B}}\|_2^3 & \text{if } \lambda_{min}^{l^3,\mathcal{B}}(T) \leq 0 \\ \frac{\lambda_{min}^{l^3,\mathcal{B}}(T)}{6} n^{-1/2} \|x - x_{\mathcal{B}}\|_2^3 & \text{if } \lambda_{min}^{l^3,\mathcal{B}}(T) > 0 \end{cases} \end{aligned}$$

using (C.1), (C.3) and (C.4) respectively. □

Bibliography

[1] C. S. Adjiman, S. Dallwig, C. A. Floudas, and A. Neumaier. A Global Optimization Method, αbb, for General Twice-Differentiable Constrained NLPs: I - Theoretical Advances. *Computers and Chemical Engineering*, 22(9):1137–1158, 1997.

[2] Ioannis G. Akrotirianakis and Christodoulos A. Floudas. A New Class of Improved Convex Underestimators for Twice Continuously Differentiable Constrained NLPs. *Journal of Global Optimization*, 30(4):367–390. ISSN 1573-2916. doi: 10.1007/s10898-004-6455-4. URL `http://dx.doi.org/10.1007/s10898-004-6455-4`.

[3] K. Anstreicher. Recent advances in the solution of quadratic assignment problems. *Math. Programming*, 97:27–42, 2003.

[4] P.I. Barton and P. Kesavan. Generalized branch-and-cut framework for mixed-integer nonlinear optimization problems. *Computers & Chemical Engineering*, 24:1361–1366, 2000.

[5] R.W. Becker and G.V. Lago. A global optimization Algorithm. In *Proc. 8th Allerton Conf. Cicuits Systems Theory (Monticello, Illinois)*, pages 3–12, 1970.

[6] Pietro Belotti, Jon Lee, Leo Liberti, Francois Margot, and Andreas Wachter. Branching and Bounds Tightening techniques for Non-convex MINLP. *Optimization Methods Software*, 24(4-5):597–634, August 2009. ISSN 1055-6788. doi: 10.1080/10556780903087124. URL `http://dx.doi.org/10.1080/10556780903087124`.

[7] C.G.E. Boender and H.E. Romeijn. Stochastic methods. In R. Horst and P.M. Pardalos, editors, *Handbook of Global Optimization*, page 829–869. Kluwer Academic Publishers, Dordrecht / Boston / London, 1995.

[8] Alberto Caprara and Marco Locatelli. Global optimization problems and domain reduction strategies. *Mathematical Programming*, 125(1):123–137, 2009. ISSN 1436-4646. doi: 10.1007/s10107-008-0263-4. URL `http://dx.doi.org/10.1007/s10107-008-0263-4`.

[9] Alberto Caprara, Marco Locatelli, and Michele Monaci. Theoretical and computational results about optimality-based domain reductions. *Computational Optimization and Applications*, pages 1–21, 2016. ISSN 1573-2894. doi: 10.1007/s10589-015-9818-5. URL `http://dx.doi.org/10.1007/s10589-015-9818-5`.

[10] Coralia Cartis, JaroslavM. Fowkes, and NicholasI.M. Gould. Branching and bounding improvements for global optimization algorithms with lipschitz continuity properties. *Journal of Global Optimization*, 61(3):429–457, 2015. URL `http://dx.doi.org/10.1007/s10898-014-0199-6`.

[11] T.G. Crainic, B. Le Cun, and C. Roucairol. Parallel branch-and-bound algorithms. In E. Talbi, editor, *Parallel Combinatorial Optimization*, pages 1–28. Wiley Series on Parallel And Distributed Computing, 2006. URL `http://books.google.co.uk/books?id=rYtuk_sm23UC`.

[12] G.B Dantzig. *Linear programming and extensions.* Rand Corporation Research Study. Princeton Univ. Press, Princeton, NJ, 1963. URL `http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+180926950&sourceid=fbw_bibsonomy`.

[13] G.B. Dantzig, S. Johnson, and W. White. A linear programming approach to the chemical equilibrium problem. *Management Science*, 5:38–43, 1958.

[14] G.B. Dantzig, S. Johnson, and W. White. On the significance of solving linear programming problems with some integer variables. *Econometrica*, 28:30–44, 1960.

[15] I. Diener. Global Optimization – Deterministic Approaches. In R. Horst and P.M. Pardalos, editors, *Handbook of Global Optimization*, pages 649–668. Kluwer Academic Publishers, Dordrecht / Boston / London, 1995.

[16] K.A. Dill, A.T. Phillips, and J.B. Rosen. Molecular structure prediction by global optimization. In I.M. Bomze, T. Csendes, R. Horst, and P.M. Pardalos, editors, *Developments in Global Optimization*, pages 217–234. Kluwer Academic Publishers, Dordrecht / Boston / London, 1997.

[17] T.G.W. Epperly and E.N. Pistikopoulos. A reduced space branch and bound algorithm for global optimization. *Journal of Global Optimization*, 11:287–311, 1997.

[18] Y. Evtushenko and M. Posypkin. An Application of the Nonuniform Covering Method to Global Optimization of Mixed Integer Nonlinear Problems. *Computational Mathematics and Mathematical Physics*, 51(8):1286–1298, 2011. URL `http://dx.doi.org/10.1134/S0965542511080082`.

[19] Y. Evtushenko and M. Posypkin. A Deterministic Approach to Global Box-Constrained Optimization. *Optimization Letters*, 7(4):819–829, 2013. URL `http://dx.doi.org/10.1007/s11590-012-0452-1`.

[20] J.E. Falk and R.M. Soland. An algorithm for separable nonconvex programming. *Management Sci.*, 15:550–569, 1969.

[21] C.A. Floudas. Deterministic Global Optimization in Design, Control, and Computational Chemistry. In Lorenz T. Biegler, Thomas F. Coleman, Andrew R. Conn, and Fadil N. Santosa, editors, *Large-Scale Optimization with Applications*, volume 93 of *The IMA Volumes in Mathematics and its Applications*, pages 129–184. Springer New York, 1997. URL `http://dx.doi.org/10.1007/978-1-4612-1960-6_7`.

[22] C.A. Floudas. *Deterministic Global Optimization: Theory, Methods and Applications*. Nonconvex Optimization and Its Applications. Springer, 1999. URL `http://books.google.co.uk/books?id=qZSpq27Ts0cC`.

[23] C.A. Floudas, I.P. Androulakis, and C.D. Maranas. *alpha*BB: A global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7(4):337–363, 1995.

[24] W. Forster. Homotopy methods. In R. Horst and P.M. Pardalos, editors, *Handbook of Global Optimization*, page 669–750. Kluwer Academic Publishers, Dordrecht / Boston / London, 1995.

[25] J.M. Fowkes, N.I.M. Gould, and C.L. Farmer. A Branch and Bound Algorithm for the Global Optimization of Hessian Lipschitz Continuous Functions. *Journal of Global Optimization*, 56(4):1791–1815, 2013. URL `http://dx.doi.org/10.1007/s10898-012-9937-9`.

[26] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[27] B. Gendron and T.G. Crainic. Parallel Branch-and-Bound Algorithms: Survey and Synthesis. *Operations Research*, 42(6):1042–1066, 1994. URL `http://dx.doi.org/10.1287/opre.42.6.1042`.

[28] Fred Glover and Manuel Laguna. Tabu search. In Panos M. Pardalos, Ding-Zhu Du, and Ronald L. Graham, editors, *Handbook of Combinatorial Optimization*, pages 3261–3362. Springer New York, 2013. URL `http://dx.doi.org/10.1007/978-1-4419-7997-1_17`.

[29] Jens Hichert, Armin Hoffmann, and HoàngXuân Phú. Convergence Speed of an Integral Method for Computing the Essential Supremum. In Immanuel M. Bomze, Tibor Csendes, Reiner Horst, and Panos M. Pardalos, editors, *Developments in Global Optimization*, volume 18 of *Nonconvex Optimization and Its Applications*, pages 153–170. Springer US, 1997. URL `http://dx.doi.org/10.1007/978-1-4757-2600-8_10`.

[30] J. Holland. Genetic algorithms and the optimal allocation of trials. *SIAM J. Computing*, 2:88–105, 1973.

[31] R. Horst and H. Tuy. *Global Optimization – Deterministic Approaches*. Springer, Berlin / Heidelberg / New York, 3rd edition, 1996.

[32] W. Kahan. A more complete interval arithmetic. Lecture notes for an engineering summer course in numerical analysis, University of Michigan, 1968.

[33] R.B. Kearfott. *Rigorous Global Search: Continuous Problems*. Kluwer Academic Publishers, Dordrecht / Boston / London, 1996.

[34] D. Knuth. *The Art of Computer Programming: Sorting and Searching*, volume 3. Addison-Wesley, 1998. URL `http://books.google.co.uk/books?id=ePzuAAAAMAAJ`.

[35] A.H. Land and A.G. Doig. An automated method for solving discrete programming problems. *Econometrica*, 28:497–520, 1960.

[36] A.V. Levy and S. Gomez. The tunneling method applied for the global minimization of functions. In P.T. Boggs, editor, *Numerical Optimization*, pages 213–244. SIAM, Philadelphia, Pennsylvania, 1985.

[37] L.-H. Lim. Singular Values and Eigenvalues of Tensors: A Variational Approach. *Computational Advances in Multi-Sensor Adaptive Processing, 2005 1st IEEE International Workshop*, pages 129–132, 2005. URL `http://dx.doi.org/10.1109/CAMAP.2005.1574201`.

[38] J.D. Little, K.C. Murty, D.W. Sweeney, and C. Karel. An algorthm for the travelling salesman problem. *Operations Research*, 11:972–989, 1963.

[39] Marco Locatelli and Fabio Schoen. *Global Optimization: Theory, Algorithms, and Applications*. SIAM, 2013. ISBN 1611972663, 9781611972665.

[40] H.M. Markowitz and A.S. Manne. On the solution of discrete programming problems. *Econometrica*, 25:84–110, 1957.

[41] G.P. McCormick. Converting general nonlinear programming problems to separable nonlinear programming problems. *Technical Report T-267, George Washington University, Washington, DC*, 1972.

[42] C.M. McDonald and C.A. Floudas. Global optimization for the phase and chemical equilibrium problem: application to the NRTL equation. *Chemical Engineering Journal*, 19:1111–1139, 1995.

[43] C.A. Meyer and C.A. Floudas. Convex Underestimation of Twice Continuously Differentiable Functions by Piecewise Quadratic Perturbation: Spline [alpha]BB Underestimators. *Journal of Global Optimization*, 32:221–258, 2005.

[44] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, Berlin / Heidelberg / New York, 3rd edition, 1996.

[45] J. Mockus, W. Eddy, A. Mockus, L. Mockus, and G. Reklaitis. *Bayesian Heuristic Approach to Discrete and Global Optimization*. Springer US, 1996.

[46] R.E. Moore. *Interval Arithmetic and Automatic Error Analysis in Digital Computing*. PhD thesis, Appl. Math. Statist. Lab. Rep. 25, Stanford University, Stanford, CA, 1962.

[47] R.E. Moore and C.T. Yang. Interval analysis I. Technical Report Space Div. Report LMSD285875, Lockheed Missiles and Space Co., 1959.

[48] Jorge J. Moré and Zhijun Wu. Global Continuation for Distance Geometry Problems. *SIAM Journal on Optimization*, 7(3):814–836, 1997. URL `http://dx.doi.org/10.1137/S1052623495283024`.

[49] T.S. Motzkin and E.G. Strauss. Maxima for graphs and a new proof of a theorem of Turan. *Canad. J. Math.*, 17:533–540, 1965.

[50] A. Neumaier. *Interval Methods for Systems of Equations*. University Press, Cambridge, 1990.

[51] A. Neumaier. Molecular modeling of proteins and mathematical prediction of protein structure. *SIAM Review*, 39:407–460, 1997.

[52] A. Neumaier. *Constraint satisfaction and global optimization in robotics, Manuscript*. 2003.

[53] A.R.F. O'Grady, I.D.L. Bogle, and E.S. Fraga. Interval analysis in automated design for bounded solutions. *Chemicke Zvesti*, 55(6):376–381, 2001.

[54] I.H. Osman and J.P. Kelly. *Meta-Heuristics: Theory and Applications*. Kluwer Academic Publishers, Dordrecht / Boston / London, 1996.

[55] J.D. Pintér. *Global Optimization in Action (Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications)*. Kluwer Academic Publishers, Dordrecht / Boston / London, 1996.

[56] S.A. Piyavskii. An Algorithm for Finding the Absolute Extremum of a Function. *USSR Comput. Math. and Math. Phys.*, 12:57–67, 1972.

[57] L. Qi. Eigenvalues of a Real Supersymmetric Tensor. *Journal of Symbolic Computation*, 40(6):1302–1324, 2005. URL `http://www.sciencedirect.com/science/article/pii/S0747717105000817`.

[58] H. Ratschek and J. Rokne. *New Computer Methods for Global Optimization*. Chichester, Ellis Horwood, 1988.

[59] H.S. Ryoo and N.V. Sahinidis. Global optimization of nonconvex NLPs and MINLPs with applications in process design. *Computers & Chemical Engineering*, 19(5):551–566, 1995.

[60] H.S. Ryoo and N.V. Sahinidis. A branch-and-reduce approach to global optimization. *Journal of Global Optimization*, 8(2):107–138, 1996.

[61] N.V. Sahinidis. BARON: A general purpose global optimization software package. *Journal of Global Optimization*, 8:201–205, 1996.

[62] Oleg Shcherbina, Arnold Neumaier, Djamila Sam-Haroud, Xuan-Ha Vu, and Tuan-Viet Nguyen. *Global Optimization and Constraint Satisfaction: First International Workshop on Global Constraint Optimization and Constraint Satisfaction, COCOS 2002, Valbonne-Sophia Antipolis, France, October 2002. Revised Selected Papers*, chapter Benchmarking Global Optimization and Constraint Satisfaction Codes, pages 211–222. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003. ISBN 978-3-540-39901-8. doi: 10.1007/978-3-540-39901-8_16. URL `http://dx.doi.org/10.1007/978-3-540-39901-8_16`.

[63] S. Skelboe. Computation of rational interval functions. *BIT*, 14:87–95, 1974.

[64] E.M.B. Smith. *On the Optimal Design of Continuous Processes*. PhD thesis, College of Science, Technology and Medicine, University of London, October 1996.

[65] E.M.B. Smith and C.C. Pantelides. Global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 21:S791–S796, 1997.

[66] E.M.B. Smith and C.C. Pantelides. A symbolic reformulation/spatial branch-and-bound algorithm for the global optimisation of nonconvex MINLPs. *Computers & Chemical Engineering*, 23:457–478, 1999.

[67] M. Teschner, B. Heidelberger, M. Mueller, D. Pomeranets, and M. Gross. Optimized Spatial Hashing for Collision Detection of Deformable Objects. In *Proceedings of Vision, Modeling, Visualization VMV*. 2003.

[68] A. Torn. *Global Optimization as a Combination of Global and Local Search.* PhD thesis, Abo Akademi University, HHAAA 13, 1974.

[69] R. Vaidyanathan and M. El-Halwagi. Global optimization of nonconvex MINLPs by interval analysis. In I.E. Grossmann, editor, *Global Optimization in Engineering Design*, page 175–193. Kluwer Academic Publishers, Dordrecht, 1996.

[70] S. Voss, S. Martello, I.H. Osman, and C. Roucairol. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization.* Kluwer Academic Publishers, Dordrecht / Boston / London, 1999.

[71] J.M. Zamora and I.E. Grossmann. A branch and contract algorithm for problems with concave univariate, bilinear and linear fractional terms. *Journal of Global Optimization*, 14:217:249, 1999.

[72] Q. Zheng and D. Zhuang. Integral global minimization: algorithms, implementations and numerical tests. *Journal of Global Optimization*, 7:421–454, 1995.

[73] A.A. Zhigliavsky and J Pintér. *Theory of global random search.* Dordrecht [Netherlands] ; Boston : Kluwer Academic Publishers, 1991.

[74] J. Zilinskas and I.D.L. Bogle. Evaluation ranges of functions using balanced random interval arithmetic. *Informatica*, 14(3):403–416, 2003.