RICE UNIVERSITY

Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems

by

Mark Aaron Abramson

A Thesis Submitted
in Partial Fulfillment of the
Requirements for the Degree

Doctor of Philosophy

Approved, Thesis Committee:

Dr. John E. Dennis, Jr., Co-Chair
Noah G. Harding Professor Emeritus
and Research Professor
Computational and Applied
Mathematics

Dr. Charles Audet, Co-Chair
Assistant Professor
Départment de Mathématiques et de
Génie Industriel
École Polytechnique de Montréal

Dr. Yin Zhang
Associate Professor
Computational and Applied
Mathematics

Dr. Andrew Meade
Associate Professor
Mechanical Engineering and Material
Science

Dr. Amr El-Bakry
Adjunct Associate Professor
Computational and Applied
Mathematics

Dr. Richard Carter
Adjunct Professor
Computational and Applied
Mathematics

Houston, Texas

August 2002

ABSTRACT

Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems

by

Mark Aaron Abramson

A new class of algorithms for solving nonlinearly constrained mixed variable optimization problems is presented. The Audet-Dennis Generalized Pattern Search (GPS) algorithm for bound constrained mixed variable optimization problems is extended to problems with general nonlinear constraints by incorporating a filter, in which new iterates are accepted whenever they decrease the incumbent objective function value or constraint violation function value. Additionally, the algorithm can exploit any available derivative information (or rough approximation thereof) to speed convergence without sacrificing the flexibility often employed by GPS methods to find better local optima. In generalizing existing GPS algorithms, the new theoretical convergence results presented here reduce seamlessly to existing results for more specific classes of problems. While no local continuity or smoothness assumptions are made, a hierarchy of theoretical convergence results is given, in which the assumptions dictate what can be proved about certain limit points of the algorithm. A new Matlab$^{\copyright}$ software package was developed to implement these algorithms. Numerical results are provided for several nonlinear optimization problems from the CUTE test set, as well as

a difficult nonlinearly constrained mixed variable optimization problem in the design of a load-bearing thermal insulation system used in cryogenic applications.

Acknowledgments

I express my sincere appreciation to all those who have helped me through this very challenging ordeal within the time constraints the U.S. Air Force has placed on me. I wish I could list everyone by name, but it could probably fill several pages.

I owe a tremendous debt of gratitude to my advisor, Professor John Dennis, whose talents extend well beyond his field of expertise into the realm of guidance, motivation, and leadership. I also thank Professor Charles Audet, who is technically my co-advisor, even though he lives in Canada. His expertise has been equally helpful, especially with getting certain mathematical details right.

I would also like to thank the rest of my committee, Professors Yin Zhang, Andrew Meade, Amr El-Bakry, and Richard Carter, who have all given me sound advice and help at one time or another during this process, and who did a wonderful job providing me feedback on this document. In particular, Professor Meade was helpful in guiding me through an engineering problem whose physics I didn't understand at first.

I thank Dr. Robert Mohling of Technology Applications, Incorporated, for his assistance in obtaining some badly needed and hard-to-find engineering materials data, and also my friend and fellow student, Keith Berrier, for his invaluable help in developing and improving my Matlab® software, saving me perhaps months of headaches and frustration.

Special thanks are warranted for Daria Lawrence and Ginger Wright (who really run the Rice CAAM Department) for always keeping me straight with respect to administrative issues. I think that together there isn't a question they can't answer.

Most importantly, I thank my wife and six wonderful children, who the U.S. Air Force prevents me from naming, due to an absurd new policy. Their support, patience,

and sacrifice has been wonderful and exemplary. This is partially their work because I could not have accomplished this without them.

Finally, I'd like to thank my father, Robert Paul Abramson (1934–2001), who taught me to love learning and to love mathematics, and my mother (name omitted due to USAF policy), whose love and gentle strength has been truly inspirational. This work is dedicated to them.

Contents

Illustrations

Tables

Chapter 1

Introduction

1.1 Mixed Variable Optimization Problems

Optimization problems arise in engineering when the goal is to find a design, expressed as a vector of variables, that minimizes or maximizes some function that acts as a measure of the merit of the design. Typically, there are also constraints on the variables, expressed as inequalities, to limit the choices of values the variables are permitted to take on. One of the broadest classes of optimization problems is known as mixed-integer nonlinear programming (MINLP) problems, which are characterized by nonlinear objective and constraint functions with a mixture of integer and continuous variables. An MINLP problem without discrete variables is simply a nonlinear programming (NLP) problem. Similar to MINLP problems is a more general class of problems only recently studied rigorously, known as mixed variable programming (MVP) problems, in which some of the discrete variables are categorical.

Categorical variables are those that take on values from a pre-defined list of categories. Some common examples are color, shape, or type of material. They can be (and often are) assigned discrete values, but the values usually have no inherent ordering, and thus are meaningless. For example, in a structural design problem with material type as a variable, discrete values might be assigned, such as 1 = steel, 2 = aluminum, etc. When considering potential iterative solution approaches, categorical variables can be thought of as variables whose "discreteness" must be satisfied at every iteration, and not just at the solution. In many applications, these variables are parameters to a large and expensive simulation, based on differential equation

models, in which the simulation software simply rejects values that do not come from a pre-defined list.

The traditional approach to engineering design problems with categorical variables is to conduct parametric studies, in which designs are optimized for specific fixed values of the categorical variables and then compared against each other. Test values are typically chosen based on an engineer's judgement and familiarity with the problem. Problems with only one categorical variable that can take on only a few values are usually not difficult to solve, since the possible choices for the categorical variable value can be exhaustively enumerated at relatively low cost. However, problems with many categorical variables that can take on several different values can quickly become too costly to solve with this approach.

In addition to the general nature of the variables, many optimization problems do not have well-behaved objective and constraint functions. Specifically, functions may have discontinuities, undefined regions, or take on infinite value at specific points. In fact, in some engineering simulations, function calls may even result in the function simply not returning a value (*e.g.*, see Booker et al. [20]). This happens, for example, when each function evaluation requires the solution of a system of differential equations, and the method used to numerically solve the system fails to generate a solution. Classical gradient-based or Newton-based methods cannot be used unmodified on problems of this type; indeed, few existing methods can. Although the objective and constraint functions will not be assumed to have any continuity or smoothness properties, the strength of the results that can be shown both theoretically and computationally will depend on such properties.

Another challenging aspect of MVP problems is that changes in the discrete variables can mean a change in the constraints, and even in the dimensions of the problem. For example, if an engineer wants to build a structure, different materials will have

different characteristics, such as bounds on thicknesses. Such is the case in [89], where the number of continuous variables is a function of the number of heat intercepts in a thermal insulation system, the latter being treated as a categorical variable. The constraint set can change, since each continuous variable has simple bounds. In fact, in this problem, even the dimension of the categorical variable space can change, since the types of insulators between intercepts are categorical design variables, and a change in the number of intercepts changes the number of insulators. In Chapter 7, the problem is expanded from that of [89] to include nonlinear constraints on system weight, stress, and thermal contraction. If constraints were placed, for example, on the thermal contraction of each insulator, then the number of these constraints would also change, depending on the number of intercepts used.

Allowing for this flexibility is a nice feature, but it complicates the mathematical formulation of the problem. In order to fully describe it, we will partition a point x into its continuous and discrete parts x^c and x^d, respectively; *i.e.*, $x = (x^c, x^d)$, where $x^d \in \mathcal{Z}^{n^d}$ and $x^c \in \Re^{n^c}$, and where n^c and n^d denote the maximum dimensions of the continuous and discrete variables, respectively. By convention, we simply ignore unused variables.

Similarly, since the number of nonlinear constraints can vary as well, we let $p < \infty$ be the maximum number of constraints, and by convention, any unused constraints are set to the zero function, so that they are always trivially satisfied.

The MVP problem of interest in this research is expressed as follows:

$$
\begin{aligned}
&\min_{x \in X} \quad f(x) \\
&\text{s. t.} \quad C(x) \leq 0,
\end{aligned}
\tag{1.1}
$$

where $f : X \to \Re \cup \{\infty\}$, and $C : X \to (\Re \cup \{\infty\})^p$ with $C = (C_1, C_2, \ldots, C_p)^T$. The domain X is partitioned into continuous and discrete variable spaces X^c and X^d,

respectively, where $X^c \subseteq \Re^{n^c}$ and $X^d \subseteq \mathcal{Z}^{n^d}$. That is, $X = X^c \times X^d$, but constraints can be functions of both x^c and x^d.

Furthermore, X^c is defined by bound and simple linear constraints; *i.e.*, for each fixed set of discrete variable values x^d, we have

$$X^c(x^d) \quad = \quad \{x^c \in \Re^{n^c} : \ell(x^d) \le A(x^d)x^c \le u(x^d)\}, \tag{1.2}$$

where $A(x^d) \in \Re^{m^c \times n^c}$, $\ell(x^d), u(x^d) \in (\Re \cup \{\pm\infty\})^{n^c}$, and $\ell(x^d) \le u(x^d)$. We denote the entire feasible region by $\Omega = \{x \in X : C(x) \le 0\}$, with its corresponding partitioning $\Omega = (\Omega^c, \Omega^d)$.

Again, the key point here is that changes in the discrete variables values can result in changes to the constraints; hence, the explicit dependence of X^c on x^d in (1.2). Note that if $n^d = 0$, the problem reduces to a standard NLP problem, in which ℓ, A, and u do not change. For convenience, we now omit the explicit dependence on x^d in this document, even though it is understood for MVP problems.

1.2 Optimality Conditions

In order to prove certain theoretical results later, some basic optimization terminology is now provided. The concepts of tangent and normal cones are first defined, after which the Karush-Kuhn-Tucker (KKT) and Fritz John first-order necessary conditions for optimality are given. For the purpose of clarity – and in this section only, – we assume that the constraints C include the linear constraints that would normally define X, and thus, $X^c = \Re^{n^c}$. Note that this is allowed within the construction of (1.1).

It is also necessary to point out that many definitions and theorems in this document, including the description of the KKT and Fritz John conditions below, have been modified slightly so that they can be applied seamlessly to MVP problems.

Specifically we add the phrase, *with respect to the continuous variables*, to mean that a stated condition applies to the continuous variables while holding the discrete variables constant. For example, the gradient of f with respect to the continuous variables is denoted $\nabla^c f$, and a function f is differentiable at x with respect to the continuous variables if $\nabla^c f(x)$ exists.

The first definition can be found in [114].

> **Definition 1.1** A vector $w \in \Re^n$ is *tangent* to the set Y at $x \in Y$ if for all sequences $\{x_i\} \subset Y$ with $x_i \to x$, and all positive scalar sequences $t_i \downarrow 0$, there exists a sequence $w_i \to w$ such that $x_i + t_i w_i \in Y$ for all i.
>
> The *tangent cone* $T_Y(x)$ is the collection of all tangent vectors to Y at x.

If Y is a convex set (*i.e.*, if $v, w \in Y$, then $\alpha v + (1 - \alpha)w \in Y$ for all $\alpha \in [0, 1]$), then a straightforward application of Definition 1.1 yields

$$T_Y(x) = \mathrm{cl}\{t(w - x) : t \geq 0, w \in Y\}. \tag{1.3}$$

This latter definition will be used often, since X^c is a convex set.

The following definition (see [133]) will be useful in defining the normal cone and other cones described in this document.

> **Definition 1.2** The *polar* of a cone V, denoted by V°, is given by
>
> $$V^\circ = \{v \in \Re^n : v^T w \leq 0 \ \ \forall \, w \in V\}. \tag{1.4}$$

The definition of the normal cone is always as the polar (or orthogonal complement) of the tangent cone [114]:

> **Definition 1.3** The *normal cone to Y at x*, denoted by $N_Y(x)$, is given by
>
> $$N_Y(x) = T_Y^\circ(x) = \{v \in \Re^n : v^T w \leq 0 \ \ \forall \, w \in T_Y(x)\}. \tag{1.5}$$

The following theorem describes the Karush-Kuhn-Tucker (KKT) first-order necessary conditions for optimality. Originally published in [83] and [92], its proof is omitted, since it can be found in many standard nonlinear programming textbooks, such as [14]. The constraint qualification mentioned in the statement of the theorem is an additional assumption that must be satisfied in order for the theorem to hold. Several different constraint qualifications appear in the literature, and are summarized in [14] or [104]. Among these is the KKT constraint qualification, which is stated immediately preceding the theorem. It can be shown that a sufficient condition for the first-order KKT constraint qualification to hold at x is that the Jacobian of the binding constraints (*i.e.*, i, where $C_i(x) = 0$ holds) has full rank [14].

Definition 1.4 The *KKT first-order constraint qualification* holds at $x \in \Re^{n^c} \times \mathcal{Z}^{n^d}$ with respect to the continuous variables if, for every $v \in T_{\Omega^c}(x)$, there exists a continuously differentiable feasible arc $\alpha : [0, \tau] \to \Re^{n^c}$ for some real scalar $\tau > 0$ such that $\alpha(0) = x^c$ and $\alpha'(0) = v$.

Theorem 1.5 Let x be a feasible solution of the optimization problem (1.1), and let $I = \{i : C_i(x) = 0\}$. Suppose that f and C_i, $i \in I$, are differentiable at x with respect to the continuous variables, and that a constraint qualification holds with respect to the continuous variables. If x is a local optimal solution, then $\nabla^c f(x)^T v \geq 0$ for all $v \in T_{\Omega^c}(x)$, and $-\nabla^c f(x) \in N_{\Omega^c}(x)$. The point x is said to satisfy the *first-order Karush-Kuhn-Tucker (KKT) necessary conditions for optimality* with respect to the continuous variables.

An alternative to the KKT conditions for optimality is the Fritz John conditions (see [82]). These conditions also have a similar constraint qualification, but it is automatically satisfied for any problem without equality constraints, as is the case

for (1.1). The Fritz John conditions are described by the following theorem, whose proof is omitted but can be found in many textbooks, including [14] and [104].

> **Theorem 1.6** Let x be a feasible solution of the optimization problem (1.1). If f and C are differentiable at x with respect to the continuous variables, then there exist scalars $\mu_0, \mu_1, \ldots, \mu_m$, not all zero, such that $\mu_i C_i(x) = 0$ and $\mu_i \geq 0$ for all $i = 1, 2, \ldots, m$, and $\mu_0 \nabla^c f(x) = \sum_{i=1}^{m} \mu_i \nabla^c C_i(x)$. The point x is said to satisfy the *first-order Fritz John necessary conditions for optimality* with respect to the continuous variables.

We omit the discussion of other optimality conditions which provide for stronger results, since the algorithms discussed in this document are generally unable to satisfy the hypotheses needed to obtain the stronger results.

1.3   Purpose of the Research

To date, methods to solve MVP problems are limited. In Chapter 2, we will see that traditional MINLP methods cannot solve these problems, and search heuristics generally lack the rigor needed to prove desired convergence properties. Perhaps the most promising approach is the class of algorithms known as Generalized Pattern Search (GPS) methods, which will be discussed in detail in subsequent chapters. In fact, Audet and Dennis recently introduced a GPS method for solving MVP problems with simple bound constraints [8].

The purpose of this research is to develop a suitable GPS algorithm to numerically solve MVP problems with general nonlinear constraints. Specifically, the goal was to devise the algorithm, establish conditions by which a limit point satisfies certain optimality conditions, prove convergence to a limit point, implement the algorithm

in a suitable programming language (Matlab®), and demonstrate its effectiveness on a real engineering design problem.

1.4   Overview

The remainder of this document is laid out as follows. Following a review of the appropriate literature in Chapter 2, Chapter 3 summarizes important definitions and results from the theory of positive linear dependence [38] and the Clarke nonsmooth calculus [31] that will be used extensively throughout this document. Chapter 4 discusses in greater detail the basic GPS algorithm for NLP problems with bound and simple linear constraints [8, 94, 95, 130], and the filter GPS algorithm for general NLP problems [7]. Chapter 5 presents the Audet-Dennis GPS algorithm for solving bound constrained MVP problems [8], but with new convergence results (for bound and simple linear constraints), in which smoothness conditions are relaxed and made more consistent with existing GPS theory. Chapter 6 presents the new class of GPS algorithm for MVP problems with general nonlinear constraints, along with its theoretical convergence properties. The algorithm is constructed as a generalization of the algorithms presented in Chapters 4 and 5. Chapter 7 provides an overview of a new Matlab® implementation of the entire class of GPS algorithms discussed here, and presents results when applied to a problem in the design of a load bearing thermal insulation system. Chapter 8, which primarily contains the work found in [2], presents a version of the basic GPS algorithm that uses any available derivative information to reduce the number of function evaluations for each GPS iteration. It includes numerical results on some test problems from the CUTE [18] set. Finally, Chapter 9 offers concluding remarks and several recommendations for future research.

Chapter 2

Literature Survey

In considering the solution of MVP problems, a discussion of the existing literature is warranted. Specifically, three relevant classes of solution approaches will be addressed in this chapter; namely, relaxation methods for solving MINLP problems, search heuristics, and pattern search methods. For each class of methods, strengths and weaknesses will be explored, relative to the type of problems we want to solve. At the end of this discussion, we offer a summary of why generalized pattern search methods are a logical approach for solving MVP problems with nonlinear constraints.

2.1 Methods for Solving MINLP Problems

Methods that have been developed for solving MINLP problems all involve iteratively solving subproblems which are relaxed in some way. Since these relaxations involve, at some point, relaxing the integrality of the integer variables, they cannot be used to solve MVP problems. Despite this drawback, a brief overview of each class of these methods is included for the sake of completeness. Classes of MINLP methods include Outer Approximation, Generalized Benders Decomposition, Branch and Bound, and the Extended Cutting Plane method.

2.1.1 Outer Approximation

Outer Approximation was first introduced for a class of MINLP problems by Duran and Grossman [43] and then extended to more general problems by Fletcher and Leyffer [46]. At each iteration, an upper bound is obtained by solving a restricted

NLP, in which discrete variables are held constant. Then, if the problem is convex (*i.e.*, the objective is convex and the constraints form a convex set), a lower bound is obtained by solving a mixed integer linear program (MILP), in which linearizations of the objective and constraint functions (at the current iterate) are cumulatively added as constraints (*i.e.*, a constraint added in one iteration is kept for all subsequent iterations). At each successive step, the lower and upper bounds approach each other, yielding an approximate solution. Variations and improvements of the method have been proposed by Kocis and Grossman [88], Leyffer [97], Floudas [50], and Viswanathan and Grossman [135].

2.1.2 Generalized Benders Decomposition

Similar to outer approximation, Generalized Benders Decomposition, developed by Geoffrion [56], solves the same NLP subproblem, but a different MILP subproblem, which is formed by linearizing the Lagrangian function $L(x, \lambda) = f(x) + \lambda^T C(x)$ around the current point, where $\lambda \in \Re^p$ is the vector of Lagrange multipliers.

2.1.3 Branch and Bound Methods

Branch and Bound methods were introduced by Dakin [36], and further developed by Gupta and Ravindran [65], Borchers and Mitchell [22], and Leyffer [98]. In this class of methods, if a continuous NLP relaxation of the MINLP does not produce a feasible solution (*i.e.*, integer variables take on integer values), then the NLP solution is treated as a lower bound, and a binary tree search is performed by implicit enumeration, where a subset of the discrete variables is fixed at each node. For example, for the discrete variable x at a particular node, if an iteration yields $x = 3.5$, the two branching problems emanating from that node would typically add the constraints, $x \leq 3$ and $x \geq 4$, respectively. Several efficiencies are added to prevent testing unnec-

essary nodes (fathoming). Quesada and Grossman [118] propose a clever LP/NLP based branch and bound method for a subclass of MINLPs, in which discrete variables are binary, while Leyffer [98] proposes a basic sequential quadratic programming (SQP) branch and bound algorithm.

2.1.4 Extended Cutting Plane Method

The Extended Cutting Plane method was introduced by Westerlund and Pettersson [136] as an extension of Kelley's [84] cutting plane algorithm for NLPs. In this algorithm, a non-decreasing sequence of lower bounds is generated by solving a sequence of MILPs, in which each MILP adds as a constraint a linearization of the most violated MINLP constraint evaluated at the previous suboptimal point. Termination with a solution occurs when the maximum constraint violation falls below a user-defined threshold.

2.1.5 Suitability of MINLP Methods

All of the current methods for solving MINLP problems have drawbacks that preclude their use on MVP problems. First, objective and constraint functions are often not convex, a condition on which convergence results for these methods greatly rely. In fact, while heuristics exist to alleviate problems with non-convexities, no local convergence theory has been been developed for non-convex problems. Second, methods that linearize objective, constraint, or Lagrangian functions make use of first-order Taylor Series, which requires differentiability of those functions; this cannot be guaranteed here. Some of these methods also require good estimates of Lagrange multipliers, which can be problematic. Third (and most importantly), all of these methods rely on some form of integer relaxation, which cannot be applied to categorical variables, since the functions are not defined outside the domain of the variables. For example,

a simulation that computes function values may only allow a fixed set of input values for each categorical variable and no others.

2.2 Search Heuristics

Search heuristics are methods designed to find global optima without using derivative information by methodically searching the solution space. Many are inspired by physical or biological processes, but lack the necessary theory to guarantee convergence to a solution satisfying first-order optimality conditions. The classes of search heuristics most relevant to solving MVP problems are simulated annealing, tabu search, and evolutionary algorithms, each of which is now discussed.

2.2.1 Simulated Annealing

Simulated annealing is a modified Monte Carlo search technique originally devised by Metropolis *et al.* [109], as an approach to numerically computing equations of state and other properties of interacting molecules. In an annealing process, a melt of a substance begins at high temperature and is slowly cooled while trying to maintain thermodynamic equilibrium. If done too quickly, or if the starting temperature is too low, then deformations of the material or other undesirable things may occur, leading to a less-than-optimal result, which is analogous to attainment of a local minimum, rather than a global minimum. Metropolis *et al.* attempted to minimize the energy $E : \Re^n \to \Re$ of a system as it is cooled to its final temperature $T = 0$ (or in the general case, $T = T_{\min}$ for some minimum temperature $T_{\min}$), essentially by the algorithm shown in Figure 2.1 with T decremented in accordance with a user-specified *cooling schedule*.

Simulated annealing was formally introduced as a general combinatorial optimization technique by Kirkpatrick, Gellatt, and Vecchi [87], and by Cerny [29] indepen-

<div style="border: 1px solid black;">

Simulated Annealing Algorithm

Choose initial point $x \in \Re^n$ at random

While $(T > T_{\min})$ do

- Randomly generate a neighbor y of x
- If $(E(y) < E(x))$, then
    - Set $x = y$

  Else

    - Set $x = y$ with probability $e^{-\frac{E(y)-E(x)}{T}}$
    - Decrease T slightly (according to the cooling schedule)

  End

</div>
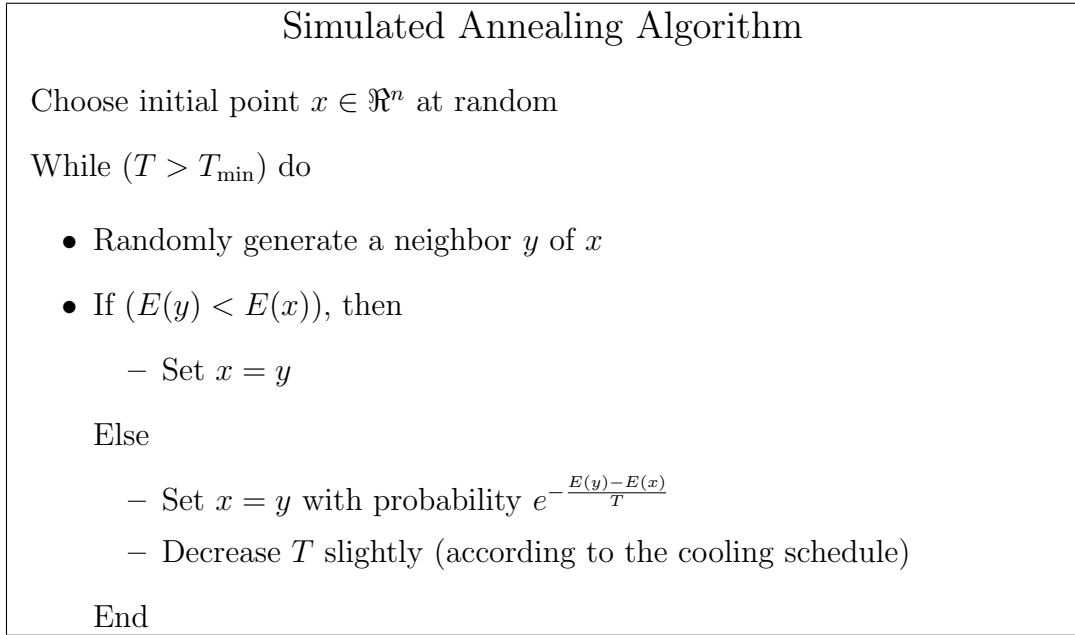
Figure 2.1   A Simulated Annealing Algorithm

dently. Much of the theory for combinatorial optimization problems relies on the asymptotic behavior of Markov chains (e.g., see [55], [57], [102], [111], and [115]). Hajek [66] is generally given credit for first establishing necessary and sufficient conditions for proving that the algorithm converges to a global minimum in probability. Specifically, the cooling schedule must be deterministic and decrease to zero, and the state space must be finite.

In a letter to the editor of *Operations Research*, Pincus [117] first suggested the application of Metropolis' work to the minimization of real continuous functions over a compact set in $\Re^n$. Actual adaptations of the simulated annealing algorithm for continuous variable problems have been proposed by many authors [17, 127, 134]. For example, Bilbro and Snyder [17] introduce what they term *tree annealing*, in which they add a tree structure to the the original Metropolis scheme to handle the

continuous variables. Szu and Hartley [127] proved convergence in probability to a global minimum for a specific choice of cooling schedule, provided that the search directions are generated according to a Cauchy distribution. In what remains an active area of research, convergence proofs for other classes of annealing algorithms, having various assumptions on cooling schedules and probability distributions can be found in [16], [54], [100] and [101], among many others. An excellent overview of simulated annealing and description of its convergence properties is found in [131].

Although convergence in probability to the global optimum is certainly a desirable result, computational studies show that the rate of convergence is often slow [79]. A theoretical reasoning for this can be found in [102], where convergence on some combinatorial optimization problems is shown to be exponentially long. Furthermore, Bilbro and Snyder [17] and Rutenbar [121] identify problems for which simulated annealing problems are unable to find the global minimum in finite time.

Ingber [78] offers some algorithmic improvements without sacrificing theoretical convergence. His implementation shows favorable performance results compared to that of the traditional simulated annealing [79] and genetic algorithms [80].

2.2.2   Tabu Search

The tabu search was formally introduced by Glover [59, 60, 61] as a metaheuristic for solving combinatorial optimization problems, although several of its ideas were developed independently by Hansen [67]. The basic idea is that, in searching for a better point among its discrete neighbors (defined by the user), the algorithm may accept a worse point if no better ones are found, and if the candidate point is not already on a list of forbidden points (called a *tabu* list). Thus, if a local optimum is found, it is added to the tabu list, and the algorithm moves to another point in an area of the domain that has not been searched yet. The tabu list is designed to

prevent the algorithm from returning back to local minima. Key to this approach is the management of the tabu list, and the strategies for three concepts, known as *aspiration*, *diversification*, and *intensification*. Aspiration refers to a function and conditions by which the tabu list is overridden to include good points. Diversification and intensification refer to strategies for searching globally or locally, respectively. Variations of tabu search differ primarily in these three areas, as well as the data structures used for managing the tabu list.

A simple tabu search procedure is given in Figure 2.2, where we define the set $N(x)$ as the discrete set of feasible neighbors at x, the set T as the tabu list, $k_{\max}$ as the maximum number of iterations, and L as the maximum number of iterations since the incumbent solution x^* was last updated.

Tabu Search Algorithm

Select an initial $x \in \Omega$ and let $x^* = x$, $T = \emptyset$

For $k = 1, 2, \ldots$

- If $(N(x) \setminus T \neq \emptyset)$, then
  - Set $s^* \in \arg\min\{f(s) : s \in N(x) \setminus T\}$
  - Set $x^* \in \arg\min\{f(s^*), f(x^*)\}$
- If $(k > k_{\max}$ or $k > L$ or $N(x) \setminus T = \emptyset)$, STOP
- Update T
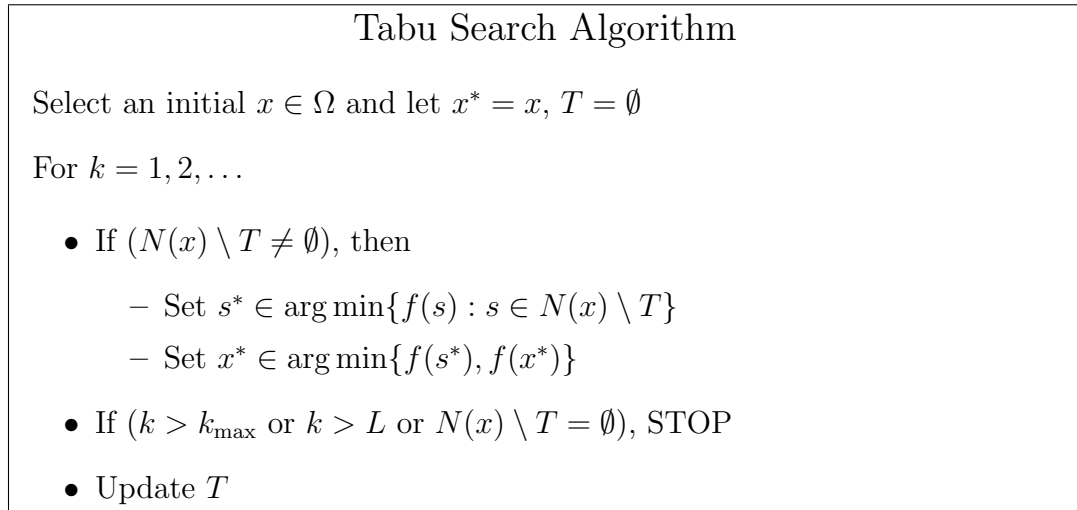
Figure 2.2 A Tabu Search Algorithm

Glover [62] offers ideas for extending the tabu search heuristic to nonlinear and parametric optimization problems. He points out that the tabu list must incorporate some notion of distance between the current point and the points in the list. The

ideas he proposes mainly involve directional search strategies and scatter search. Although tabu search is widely used and has performed well on many combinatorial optimization problems in practice, there is no formal convergence theory to guarantee its success.

2.2.3   Evolutionary Algorithms

Evolutionary algorithms encompass a large family of numerical methods that model the evolutionary processes seen in biology. Among these are three classes of algorithms that have been applied successfully to optimization problems; namely, Evolution Strategies, Evolutionary Programming, and Genetic Algorithms. The latter, which are the most well-known, have been primarily used on discrete optimization problems, while the other two were designed primarily for continuous problems. Genetic algorithms have been applied to continuous problems as well, but to do so generally requires a discrete encoding of the problem.

Rather than dealing with one iterate at a time, evolutionary algorithms deal simultaneously with finitely many points, called a *population*. Each point or *individual* in the population is evaluated with respect to a *fitness function*, which is analogous to an objective function. Each *generation* then reproduces when a subset of individuals in the population, called *parents*, are selected based on some specified criteria, and various search operators are applied to the parents to create new individuals, called *offspring*. Typically, the selection and reproduction processes ensure that offspring, on average (*i.e.*, expected value), will have better fitness values than the parents; however, this is not guaranteed, due to the randomness inherent in the reproduction process.

The search operators are typically given biological names, the most common of which are selection, reproduction (often called recombination or crossover), muta-

tion, and competition [122]. They are the tools by which new candidate points are generated in the optimization process. These can be described briefly as follows:

- **Selection:** Process by which parents are selected for reproduction.

- **Reproduction:** Process by which genes are passed from parents to offspring.

- **Mutation:** Random errors occurring in the reproduction process.

- **Competition:** Process by which offspring survive to the next generation when there are limited resources.

Given these definitions, it is helpful to point out a few examples. Suppose parents $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ have been selected. One common recombination operator produces two complementary offspring $z, w \in \Re^n$, in which, $z_i = x_i, w_i = y_i$ with probability p_i; otherwise, $z_i = y_i, w_i = x_i$ [138]. Another common operator computes some linear combination of the parents' characteristics for each component. Mutation operators may involve adding some random values to components of a member of the population. The random values are typically statistically independent and come from some pre-defined probability distribution, most commonly Gaussian or Cauchy.

A general framework [138] for Evolutionary Algorithms is given in Figure 2.3, the details of which will define the various classes of algorithms.

Genetic Algorithms

Genetic Algorithms were developed by Holland [74, 75] as a model of Darwinian [37] theory of genetic evolution, where members of a population with higher fitness values are given a higher probability of reproducing. In most implementations, variables are encoded as binary strings, and the typical approach is to include the four search

---

Evolutionary Algorithm

Generate the initial population $P(0)$ at random

For $i = 0, 1, 2, \ldots$

- Evaluate the fitness of each individual in $P(i)$
- Select parents from $P(i)$
- Apply search operators to the parents and produce generation $P(i+1)$
- Check for convergence or termination condition
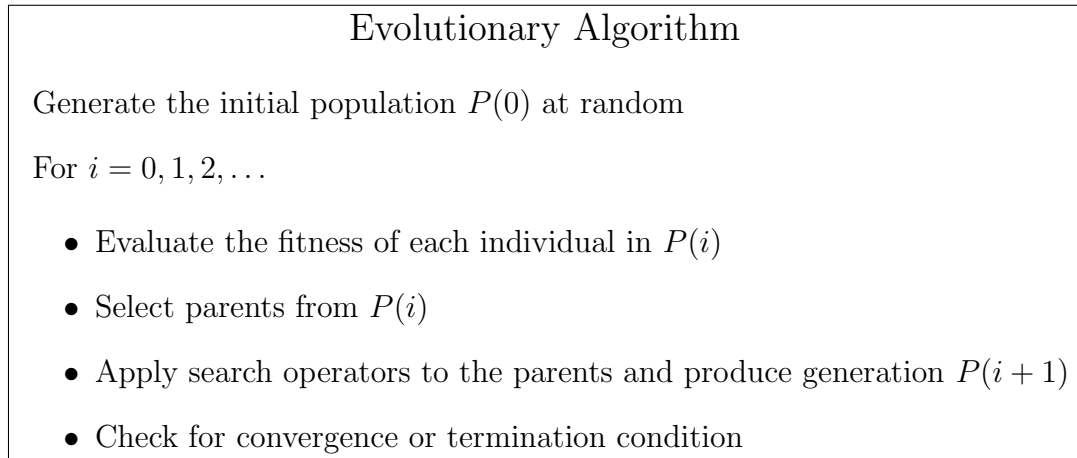
Figure 2.3   An Evolutionary Algorithm

operators in the genetic process. Genetic algorithms were first applied to optimization by De Jong [39], a student of Holland.

Key to the algorithm is the choice of fitness function. This may be the same as the objective function of the problem being optimized, but often is not, in an effort to increase efficiency. However, the extrema of the two functions must match [122].

Selection of parents for reproduction is usually done randomly, with increased probability of selection for individuals with better fitness values. The most common approaches select parents based on proportional values of the fitness functions [63], ordinal rankings of individuals by fitness value [11], or a combination of the two [63, 110].

In genetic algorithms, since the individuals are usually represented by binary strings, the crossover (reproduction) operator involves sampling bits from the two parents. One of the most common schemes generates two children by copying the parents and then swapping (proper) substrings of the binary encoding. Mutation

typically occurs with low probability and usually involves resetting a randomly selected bit to its complement.

A brief survey of convergence theory results for genetic algorithms can be found in [110]. A Markov chain analysis of genetic algorithms with finite populations is given in [64], but only considers reproduction and mutation operators. Kingdon [86] studied and attempted to characterize problems that genetic algorithms have difficulty solving, and provides some convergence results. Rudolph [120] proved that the only genetic algorithms that converge to a global optimum are those that always maintain the best solution in the population. Michalewicz [110] introduced a new class of so-called *contractive mapping* genetic algorithms, which, under certain conditions, converge (not just in probability) based on Banach's well-known contraction mapping theorem [12]. A drawback of this approach is that the contraction mapping requires an improvement of the entire population's average fitness at each iteration. If an an improved population is not found, then more iterations are performed until one is found. As the algorithm progresses, this becomes less and less likely, which slows convergence considerably.

In practice, genetic algorithms have additional drawbacks that make them unsuitable for solving the type of MVP problems investigated here. For one, while they are generally good at finding improved designs quickly, they are not reliable at finding global optima (which is their goal), and when they do converge to a local optima, it is often at a slower rate than traditional gradient or so-called *hill-climbing* methods [52, 128]. Furthermore, because of their very nature, the requirement to generate new populations rather than single iterates makes them even more costly with respect to the number of function evaluations.

Evolutionary Strategies

Rechenberg and Schwefel first proposed evolution strategies in 1965 as a numerical optimization technique, although not in its current form [124].

In evolution strategies, each individual is represented by a pair of real-valued vectors (x, σ), where x is its position in the search space, and σ is a vector of standard deviations. The mutation operator is typically performed by $x^{k+1} = x^k + N(0, \sigma)$, where $N(0, \sigma)$ is a vector of independent normally distributed random numbers with mean 0 and standard deviation σ. Since the selection of σ is highly dependent on the problem and its dimensionality, Schwefel [123] proposed a concept referred to as *self-adaptation*, in which the mutation operator is applied to σ as well as x.

The two primary selection operators are referred to as $(\lambda + \mu)$ and (λ, μ), where μ represents the number of parents and λ represents the number of offspring. After the λ offspring are generated, the μ fittest individuals are selected, either from the total $\mu + \lambda$ candidates or only the λ offspring, depending on the respectively chosen operator. Offspring that violate any constraint of the optimization problem are discarded in favor of the parents (which are feasible).

The two primary recombination operators are exactly those mentioned earlier; namely, given parents x and y, each component of one offspring is randomly chosen from the corresponding components of the two parents, while the second offspring is its complement; or an offspring's components are generated as some linear combination of the parents' components.

Convergence in probability can be proved for optimization problems with mild assumptions (including continuity of the objective function), provided that the components of σ are identical and positive [9]. However, the question as to the convergence rate of evolution strategies remain open.

Evolutionary Programming

Evolutionary Programming was first introduced as an artificial intelligence technique applied to finite state machines by Fogel [51], and it was later extended by Burgin [26, 27] and Atmar [3]. Though developed under entirely different circumstances, it is actually quite similar to evolution strategies. For example, each individual is represented by a pair of vectors (x, σ) in precisely the same way as evolution strategies. The mutation operator is also essentially identical; however, evolutionary programming has no recombination operator [10]. Other than that, the primary differences between the two approaches involve the selection and competition processes.

Rather than selecting the best offspring, evolutionary programming uses a *tournament* approach. Once the μ offspring have been generated, yielding $\lambda + \mu$ individuals, a subset of the individuals are selected uniformly at random as opponents for each individual. Then each individual receives a score corresponding to the number of opponents with a worse fitness value. The next generation's parents are chosen as the μ individuals with the highest scores.

2.2.4 Suitability of Search Heuristics

The search heuristics presented here are all useful methods in optimization when the goal is to improve a design or find a better point. In particular, they have few restrictions on the types of problems to which they can be applied. However, though the goal in many cases is to find a global optimum, adequate theory to do so is generally absent.

Tabu search is an intriguing idea for avoiding local optima, which has performed well in a variety of problems, but it is geared mainly toward discrete optimization, and there are simply no convergence guarantees. Simulated annealing, under a specific

set of assumptions, converges in probability to a global optimum, but, in practice, is often much slower than traditional methods, due to the probabilistic nature of the algorithm. Evolutionary algorithms offer little by way of theoretical convergence guarantees without becoming overly problem-dependent, except in one particular case of evolutionary strategy. However, in practice, the convergence rate is often slow (similar to simulated annealing), and early movement toward a local optimum is common. This inefficiency is magnified as the dimension of the problem is increased.

2.3 Generalized Pattern Search (GPS) Methods

Pattern search methods represent a subclass of direct search algorithms, in which the minimizer of a continuous function is sought without the use of derivatives. Among the first direct search algorithms were the well-known method of Hooke and Jeeves [76] and the simplex algorithm of Nelder and Mead [113]. At the time, these were considered heuristics with no formal convergence theory.

2.3.1 Lewis and Torczon

In an award-winning 1997 paper, Torczon [130] introduced the class of generalized pattern search (GPS) methods for solving unconstrained NLP problems, and showed that it includes coordinate search with fixed step sizes, evolutionary operation using factorial design [23], Hooke and Jeeves' algorithm [76], and the multidirectional search algorithm [42]. Without ever computing or approximating derivatives, Torczon [130] also show that if all iterates lie in a compact set and the objective function f is continuously differentiable in a neighborhood of the level set $\{x \in \Re^n : f(x) \leq f(x_0)\}$, where $x_0 \in \Re^n$ is the initial iterate, then a subsequence of GPS iterates $\{x_k\}$ converges to a point $\hat{x}$ satisfying $\nabla f(\hat{x}) = 0$.

At each iteration of Torczon's algorithm, the objective function is evaluated on a finite set of neighboring points on a carefully constructed discrete mesh, formed by considering nonnegative integer combinations of vectors that form a positive spanning set [93] (see Definition 3.2). If an improvement is found, then the new iterate is accepted and the mesh is retained or coarsened; otherwise, the mesh is refined and a new set of neighboring mesh points is evaluated. Torczon shows convergence by showing that the mesh size gets arbitrarily small. In [93], Lewis and Torczon generalize their algorithm by applying the theory of positive linear dependence of Davis [38] to reduce the worst case number of trial points at each iteration. They also introduce a heuristic, called *rank ordering*, in which, after evaluating f in directions forming a basis, they generate a new direction, based on the difference between the best and worst directions of the basis, with the expectation that, for a sufficiently fine mesh, this new direction will provide a crude estimate for the direction of steepest descent. Similar ideas, but in a slightly different context, can be found in [34].

Lewis and Torczon have extended the GPS method to solve both bound [94] and linearly constrained problems [95]. In doing so, they showed that by choosing the search directions appropriately, and if the objective function f is continuously differentiable in a neighborhood of the level set $\{x \in \Re^n : f(x) \leq f(x_0)\}$, the algorithm is guaranteed to produce a subsequence of iterates converging to a limit point $\hat{x}$ satisfying $\nabla f(\hat{x})^T(x - \hat{x}) \geq 0$ for any feasible x. Audet [4] provides several clever examples to show that many of Torczon's theoretical results cannot be relaxed.

For NLP problems with nonlinear constraints, Lewis and Torczon [96] developed a derivative-free augmented Lagrangian version of GPS. The augmented Lagrangian they use, which comes from Conn, Gould, and Toint [32], is given by

$$\Phi(x; \lambda, S, \mu) = f(x) + \sum_{i=1}^{p} \lambda_i C_i(x) + \frac{1}{2\mu} \sum_{i=1}^{p} s_{ii} C_i(x)^2, \qquad (2.1)$$

where $\lambda = (\lambda_1, \ldots, \lambda_p)^T$ is the vector of Lagrange multiplier estimates for the equality constraints $C_i(x), i = 1, 2, \ldots, p$ (inequality constraints are assumed to have had slack variables added, as appropriate), μ is a penalty parameter, and the entries s_{ii} of the diagonal matrix S are scaling factors for the constraints. Thus, in this formulation, the nonlinear (and linear) constraints are incorporated into the augmented Lagrangian, while the simple bound constraints remain as explicit constraints. The GPS algorithm described in [94] is then applied to the resulting bound constrained subproblem in an iterative fashion. However, each subproblem (denoted by j) is only solved until the mesh size parameter satisfies $\Delta_k < \delta_j$, where $\delta_j \to 0$ and $\delta_0 \ll 1$. The Lagrange multiplier estimates are updated by the same Hestenes-Powell formula as in [32]; namely,

$$\bar{\lambda}(x, \lambda, S, \mu) = \lambda + \frac{1}{\mu} SC(x), \tag{2.2}$$

which requires no derivative information.

Lewis and Torczon show that this construction, under the same assumptions as in [32], plus a mild restriction on search directions, yields an algorithm that converges to a KKT first-order stationary point. The main drawback is that no strategy for reducing the penalty parameter is given, and no numerical results are provided. Thus, the efficiency of the algorithm is, for the most part, unknown.

An asynchronous parallel version of the GPS algorithm has also been developed by Hough, Kolda and Torczon [77, 90, 91].

2.3.2 Audet and Dennis

Audet and Dennis [6] present a hierarchy of convergence results for a slightly modified, but equivalent version of GPS for bound and linearly constraints, in which the strength of the results depends on local continuity and smoothness conditions of the

objective function. By so doing, they establish Torczon's work as a corollary of theirs with much shorter and simpler proofs. This hierarchy of results is described in detail in Chapter 4.

Two additional GPS papers, which are discussed in detail in Chapters 4 and 5, are extremely relevant to, and in fact, form the foundation of this work. In the first paper [7], Audet and Dennis implement a filter method into the GPS framework to handle general nonlinear constraints. Originally introduced by Fletcher and Leyffer [47] to conveniently globalize sequential quadratic programming (SQP) and sequential linear programming (SLP), filter methods accept steps if either the objective function or an aggregate constraint violation function is reduced. Fletcher, Leyffer, and Toint [48] show convergence of the SLP-based approach to a limit point satisfying Fritz John [82] optimality conditions; they show convergence of the SQP approach to a KKT point [49], provided a constraint qualification is satisfied. However, in both cases, more than a simple decrease in the function values is required for convergence with these properties. Audet and Dennis show convergence to limit points having almost the same characterization as in [6], but with only a simple decrease in the objective or constraint violation function required. While they are unable to show convergence to a point satisfying KKT optimality conditions (and, in fact, have counterexamples [7]), in that $-\nabla f(\hat{x})$ does not necessarily belong to the normal cone, they are able to show that $-\nabla f(\hat{x})$ belongs to a larger cone containing the normal cone, the size of which depends on the choice of directions used. A richer set of directions, although more costly, spans more of the tangent cone; thus, its polar shrinks, so as to increase the likelihood of achieving a KKT point.

In the second paper [8], Audet and Dennis introduce a GPS method to solve bound constrained MVP problems under the assumption of continuous differentiability of the objective function on the neighborhood of a level set in which all iterates lie.

This algorithm is a generalization of the basic GPS algorithm in that it reduces to basic method in the absence of discrete variables. The success of the method is demonstrated in [89] on a problem in the design of thermal insulation systems, an expanded version of which is discussed in detail in Chapter 7.

A key point to the Audet-Dennis GPS algorithms is that they explicitly separate out a SEARCH step from the main POLL step within the iteration, in which any finite search strategy (including none) on the mesh may be employed, without adversely affecting the convergence theory. This flexibility lends itself quite easily to hybrid algorithms and enables the user to apply specialized knowledge of the problem. One typical implementation for difficult and computationally expensive problems is to optimize a significantly less costly surrogate function during the SEARCH step of each iteration, map the resulting point to a nearby mesh point, and compute its true function value there [21]. While the SEARCH step contributes nothing to the convergence theory of GPS (and in fact, an unsuitable SEARCH may impede performance), the use of surrogates enables the user to gain significant improvement early on in the iteration process at much lower cost.

Because problems with very expensive function evaluations are in the target class of problems we wish to solve, a discussion of the use of surrogates is warranted. Given an optimization problem in the form of (1.1), surrogate functions $s_f : X \to \Re$ and $s_C : X \to \Re^p$ are constructed based either on simplified physics, or approximations obtained by evaluating f and C at selected points, called *data sites*, $x_1, \ldots, x_r$, and interpolating or smoothing the function values. In the SEARCH step of the GPS algorithm, the surrogate optimization problem,

$$
\begin{aligned}
&\min_{x \in X} \quad s_f(x) \\
&\text{s. t.} \quad s_C(x) \leq 0,
\end{aligned}
\tag{2.3}
$$

is solved approximately using any desired method. It is not important to obtain a high level of accuracy, unless the surrogates are accurate approximations of the true functions. The surrogate solver may return multiple points x_s, which are then mapped to the mesh. The values $f(x_s)$ and $C(x_s)$ are then computed with the hope of finding an improved feasible design for the original problem. If no improvement is found, then the GPS POLL step is invoked, and more points are evaluated with respect to the original problem. The surrogate can then be recalibrated using the new points that have been evaluated. The literature contains several reasonable approaches for handling the recalibration process [25, 58, 132].

The building of surrogates often involves the use of *surfaces*, which are functions designed to fit or smooth data chosen at the data sites. Examples include kriging, responses surfaces, polynomial interpolants, and neural networks. Data site selection is often done using Latin hypercube sampling (LHS) [107, 126], orthogonal arrays (OA) [116], OA-based LHS [129], or other probability-based space filling strategy, – or alternatively, by experimental design, with the goal of obtaining a reasonable and rich sampling of the domain.

Although the relationship between surrogates and surfaces is direct, the two are not necessarily the same. Surfaces may be constructed based on the difference between the original and surrogate functions, or on the quotient of the original and surrogate functions, in which case, the surrogate would be constructed as the sum or product, respectively, of the surface and surrogate functions. One very interesting and new approach is space mapping [40], in which the relationship between an expensive fine model and an inexpensive coarse model is explored. The fine and coarse models are analogous to the original and surrogate models, respectively.

2.4 Other Relevant Methods

A few other papers in related areas are also of interest. First, a recently published paper by Coope and Price [35] introduces what they term a *grid-based* method for unconstrained NLP problems with continuously differentiable objective functions. It is similar to that of Torczon [130], but with an alternative, more flexible construction of the mesh (or grid). They show convergence to a first-order stationary point, but must rely on the assumption that the mesh size converges to zero, which Torczon is able to prove because of the stricter mesh construction. While this method does not quite fall under the GPS umbrella, the flexibility of their grid construction allows them to extend the algorithm by constructing the grid with conjugate directions, thereby achieving the classical result of finite termination on strictly convex quadratic functions [33].

Second, Bünner, Schittkowski, and van de Braak [24] solve problems in the design of surface acoustic wave filters by applying an SQP method, that has been extended to handle non-relaxable integer variables by adding a direct search component. However, the algorithm treats the direct search method as a heuristic, and has no proven convergence theory.

Finally, Hart [68, 69, 70] introduces an evolutionary pattern search algorithm that combines the ideas of evolutionary algorithms with the Torczon [130] GPS algorithm in a way that allows him to prove probabilistic analogs of Torczon's convergence results.

2.5 Summary

In this chapter, the main ideas of MINLP methods, search heuristics, and GPS methods have been discussed. MINLP methods are not suitable for solving MVP problems

because all current methods would require some form of relaxation of the integrality of the categorical variables, which is not possible. Search heuristics are also not suitable because of the general lack of sufficient convergence theory and the large number of function evaluations typically required. However, they can certainly be used within the SEARCH step of a GPS algorithm, since optimizing a surrogate function is generally inexpensive, and a high degree of accuracy is generally not required. Finally, of the methods currently available, GPS algorithms seem best suited to solve nonlinearly constrained MVP problems, due to the combination of flexibility and convergence theory these methods possess, and because of already existing GPS methods to solve both NLPs and bound constrained MVPs. In fact, the main focus of this work is to integrate the filter method used in [7] into the GPS algorithm for MVP problems described in [8] so that MVP problems with nonlinear constraints and weaker smoothness conditions can be solved.

Chapter 3

Positive Basis Theory and Clarke Calculus

Lewis and Torczon [93] show that convergence of GPS algorithms relies on the theory of positive linear dependence of Davis [38]. Audet and Dennis [6] show that the non-smooth calculus of Clarke [31] also plays a vital role in further characterizing GPS limit points. The purpose of this chapter is to provide some basic definitions and theoretical results from both of these areas that will be used throughout the chapters that follow. The final section contains some new, but straightforward results that will be helpful in some of the GPS convergence proofs in later chapters.

3.1 Positive Spanning Sets

The following terminology is due to Davis [38].

Definition 3.1 A *positive combination* of the set of vectors $V = \{v_i\}_{i=1}^r$ is a linear combination $\sum_{i=1}^r \alpha_i v_i$, where $\alpha_i \geq 0$, $i = 1, 2, \ldots, r$.

Definition 3.2 A finite set of vectors $W = \{w_i\}_{i=1}^r$ forms a *positive spanning set* for $\Re^n$, if every $v \in \Re^n$ can be expressed as a positive combination of vectors in W. The set of vectors W is said to *positively span* $\Re^n$. The set W is said to be a *positive basis* for $\Re^n$ if no proper subset of W positively spans $\Re^n$.

Davis [38] shows that the cardinality of any positive basis in $\Re^n$ is between $n + 1$ and $2n$. Common choices for positive bases include the columns of $[I, -I]$ and $[I, -e]$, where I is the identity matrix and e the vector of ones. In fact, for any basis $V \in R^{n \times n}$, the columns of $[V, -V]$ and $[V, -Ve]$ are easily shown to be positive bases.

The following key theorem, which is the motivation behind the use of positive spanning sets in GPS algorithms [93], is also due to Davis [38].

Theorem 3.3   A set D positively spans $\Re^n$ if and only if, for all nonzero $v \in \Re^n$, $v^T d > 0$ for some $d \in D$.

Proof.   (Davis [38]) Suppose D positively spans $\Re^n$. Then for arbitrary nonzero $v \in \Re^n$, $v = \sum_{i=1}^{|D|} \alpha_i d_i$ for some $\alpha_i \geq 0$, $d_i \in D$, $i = 1, 2, \ldots, |D|$. Then $0 < v^T v = \sum_{i=1}^{|D|} \alpha_i d_i^T v$, at least one term of which must be positive. Conversely, suppose D does not positively span $\Re^n$; *i.e.*, suppose that the set D spans only a convex cone that is a proper subset of $\Re^n$. Then there exists a nonzero $v \in \Re^n$ such that $v^T d \leq 0$ for all $d \in D$. ∎

It is easy to see that if $\nabla f(x)$ exists at x and is nonzero, then, by choosing $v = -\nabla f(x)$, there exists a $d \in D$, such that $\nabla f(x)^T d < 0$. Thus at least one $d \in D$ is a descent direction.

3.2   The Clarke Calculus

3.2.1   Basic Definitions

Basic terminology for the Clarke calculus is now provided. Definitions 3.4–3.8 below are classical definitions, which can be found in [31] and [133], among others. The remaining definitions are due to Clarke [31], except for Definition 3.12, which Clarke [31] attributes to Bourbaki without a citation. Although Clarke develops the more general case of a Banach space (with possibly infinite dimensions), we restrict ourselves here to $\Re^n$, since GPS methods are not designed for infinite dimensions. Thus, all points referenced in this chapter lie in $\Re^n$, unless otherwise specified.

We should also note that there are different definitions for differentiability of a function f at a point $x \in \Re^n$. The one given below in Definition 3.8 is referred to

as Gâteaux differentiability, as opposed to the traditional Fréchet type (see [119]). However, they coincide if f is Lipschitz near x [31] (see Definition 3.5 below), which is the only case we are concerned with in this work.

Definition 3.4 Let V be a subset of $\Re^n$. A function $f : V \to \Re \cup \{+\infty\}$ is said to be *lower semi-continuous* at $x \in V$ if $f(x) \leq \liminf\limits_{y \to x} f(y)$.

Definition 3.5 A function $f : Y \subseteq \Re^n \to \Re$ is said to satisfy a *Lipschitz condition* on Y if there exists a scalar $L \geq 0$ such that, for all x, x' in Y,

$$|f(x) - f(x')| \leq L\|x - x'\|. \tag{3.1}$$

The function f is said to be *Lipschitz near* $x \in Y$ if, for some $\epsilon > 0$, f satisfies a Lipschitz condition on $B(x, \epsilon)$, where $B(x, \epsilon)$ is an open ball of radius ϵ centered at x.

Definition 3.6 A function $f : Y \subset \Re^n \to \Re$ is said to be *convex* on Y if, for all $x, y \in Y$ and $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y). \tag{3.2}$$

The function f is said to be *convex near* x if it is convex on a ball of radius ϵ around x, for some $\epsilon > 0$.

Definition 3.7 Let $f : \Re^n \to R$. The *one-sided directional derivative* of f at x in the direction $v \in \Re^n$ is given by

$$f'(x; v) := \lim_{t \downarrow 0} \frac{f(x + tv) - f(x)}{t} \tag{3.3}$$

when the limit exists.

Definition 3.8 A function $f : \Re^n \to \Re$ is said to be *differentiable* at x if there exists a vector $\nabla f(x) \in \Re^n$ such that, for every $v \in \Re^n$, $f'(x; v)$ exists and equals $\nabla f(x)^T v$.

Definition 3.9 (Clarke) Let $f : \Re^n \to \Re$ be Lipschitz near a given point x. The *generalized directional derivative* of f at x in the direction v is given by

$$f^\circ(x; v) := \limsup_{y \to x, t \downarrow 0} \frac{f(y + tv) - f(y)}{t}, \tag{3.4}$$

where t is a positive scalar.

Definition 3.10 (Clarke) The *generalized gradient* of $f : \Re^n \to \Re$ at x is defined to be the set

$$\partial f(x) := \{\zeta \in \Re^n : f^\circ(x; v) \geq v^T \zeta \text{ for all } v \in \Re^n\}. \tag{3.5}$$

Definition 3.11 (Clarke) The function $f : \Re^n \to \Re$ is said to be *regular* at x if, for each $v \in \Re^n$, $f'(x; v)$ exists and coincides with $f^\circ(x; v)$.

Definition 3.12 (Bourbaki) The function $f : \Re^n \to \Re$ is said to be *strictly differentiable* at x if, for all $v \in \Re^n$,

$$\lim_{y \to x, t \downarrow 0} \frac{f(y + tv) - f(y)}{t} = \nabla f(x)^T v. \tag{3.6}$$

3.2.2 Some Theoretical Results of Clarke

At this point, we provide some of the theoretical results proved in [31]. The following results are listed without proof, since their proofs are outside the scope of this work. We assume again that all points and sets described here reside in $\Re^n$, even though most of these results are proved in [31] for general Banach spaces.

- Let f be Lipschitz near x. Then there is a neighborhood of x in which f is differentiable except on a subset Ω_f of Lebesgue measure 0.

- Let f be Lipschitz near x, and let S be any set of Lebesgue measure 0 in $\Re^n$. Then $\partial f(x) = \mathrm{co}\{\lim \nabla f(x_k) : x_k \to x, x_k \notin S \cup \Omega_f\}$, where "co" denotes the convex hull of the specified set.

- The generalized directional derivative may be obtained from the generalized gradient as follows: $f^\circ(x; v) = \max\{v^T \zeta : \zeta \in \partial f(x)\}$.

- The generalized directional derivative, as a function of direction v is finite, positively homogeneous, and subadditive. Also the relationship $f^\circ(x; -v) = (-f)^\circ(x; v)$ holds.

- If x is a minimizer of f, and f is Lipschitz near x, then $0 \in \partial f(x)$. This is a generalization of the standard 1st order necessary condition for continuously differentiable f that $\nabla f(x) = 0$.

- If f is differentiable at x, then $\nabla f(x) \in \partial f(x)$.

- When f is continuously differentiable at x, $\partial f(x)$ reduces to the singleton $\{\nabla f(x)\}$, and when f is convex near x, it coincides with the subdifferential [73, 103]; i.e., $\partial f(x) = \{\eta \in \Re^n : f(y) \geq f(x) + \eta^T(y - x) \text{ for all } y \in \Re^n\}$.

- If f is strictly differentiable at x, then f is Lipschitz near x and differentiable at x with $\partial f(x) = \{\nabla f(x)\}$.

- If f is Lipschitz near x and $\partial f(x)$ reduces to a singleton $\{\zeta\}$, then f is strictly differentiable at x and $\nabla f(x) = \zeta$.

- If f is Lipschitz near x and is convex near x, then f is regular at x.

The final result of this section is stated formally and with proof because of its importance to Section 3.3.

Theorem 3.13 A function f is Lipschitz near x, differentiable at x, and regular at x, if and only if f is strictly differentiable at x.

Proof. (Clarke [31]) Suppose f is Lipschitz near x, differentiable at x, and regular at x. Local Lipschitz continuity ensures that $f^\circ(x; v)$ exists for all $v \in \Re^n$, and regularity ensures that $f'(x; v)$ exists and equals $f^\circ(x; v)$ for all $v \in \Re^n$. Differentiability ensures that $\nabla f(x)$ exists and that $\nabla f(x)^T v = f'(x; v) = f^\circ(x; v)$ for all $v \in \Re^n$. From this relationship and the properties above, we have the following:

$$
\begin{aligned}
\liminf_{y \to x, t \downarrow 0} \frac{f(y + tv) - f(y)}{t} &= -\limsup_{y \to x, t \downarrow 0} \frac{f(y) - f(y + tv)}{t} \\
&= -\limsup_{y \to x, t \downarrow 0} \frac{f(y + tv - tv) - f(y + tv)}{t} \\
&= -f^\circ(x; -v) = -\nabla f(x)^T(-v) = \nabla f(x)^T v \\
&= \limsup_{y \to x, t \downarrow 0} \frac{f(y + tv) - f(y)}{t}.
\end{aligned}
$$

Since the limit infimum and limit supremum coincide, the limit exists and equals $\nabla f(x)^T v$. Thus, by definition, f is strictly differentiable at x.

Conversely, suppose f is strictly differentiable at x. Applying Definition 3.12 with $y = x$ establishes the differentiability of f at x.

Now suppose f is not Lipschitz near x. Then there exist sequences $\{x_k\}$ and $\{x_k'\}$ converging to x such that both lie in the open ball of radius $\frac{1}{k}$, centered at x, and $|f(x_k') - f(x_k)| > k\|x_k' - x_k\|$. Define $t_k > 0$ and $v_k \in \Re^n$ such that $x_k' = x_k + t_k v_k$ with $\|v_k\| = \frac{1}{\sqrt{k}}$. Since x_k and x_k' both converge to x, we have $t_k \downarrow 0$. Then, by substitution and a little algebra, we have

$$
\frac{|f(x_k + t_k v_k) - f(x_k)|}{t} > \sqrt{k}. \tag{3.7}
$$

Strict differentiability of f ensures that, for any $\epsilon > 0$, there exists an n_ϵ such that, for all $k \geq n_\epsilon$, we have that

$$\left\| \frac{f(x_k + t_k v_k) - f(x_k)}{t_k} - \nabla f(x)^T v_k \right\| < \epsilon \tag{3.8}$$

However, as k gets arbitrarily large, $v_k \to 0$, making it impossible for (3.8) to hold because the term $\frac{f(x_k + t_k v_k) - f(x_k)}{t_k}$ has norm exceeding $\sqrt{k}$, which becomes unbounded. Therefore, f is Lipschitz near x.

To show that f is regular at x, we have by Definitions 3.8, 3.9, and 3.12 that for any $v \in \Re^n$,

$$
\begin{aligned}
f'(x; v) = \nabla f(x)^T v &= \lim_{y \to x, t \downarrow 0} \frac{f(y + tv) - f(y)}{t} \\
&= \limsup_{y \to x, t \downarrow 0} \frac{f(y + tv) - f(y)}{t} = f^\circ(x; v),
\end{aligned}
$$

where the third equality is due to the fact the limit exists and therefore equals the limit supremum. $\blacksquare$

3.2.3   Examples

A few examples are now provided to illustrate the Clarke calculus concepts described in Section 3.2. The first example, taken from Clarke [31], is the simple absolute value function, and is included to simply illustrate the basic definitions.

Example 3.14   Consider the function $f : \Re \to \Re$ defined by $f(x) = |x|$. For $x > 0$, the definitions yield, $f^\circ(x; v) = v$ and $\partial f(x) = \{1\}$, while, for $x < 0$, we get $f^\circ(x; v) = -v$ and $\partial f(x) = \{-1\}$. These results are entirely consistent with the classical results that $f'(x) = 1$ for $x > 0$, and $f'(x) = -1$ for $x < 0$. However, for $x = 0$, it is clear that f is not differentiable; hence, it is neither strictly nor continuously differentiable.

A straightforward application of the definition yields $f°(0, v) = |v|$, and f is regular, since $f°(0; 1) = 1 = f'(0; 1)$ and $f°(0; -1) = -1 = f'(0; -1)$. By definition, $\partial f(0)$ contains all ζ satisfying $|v| \geq \zeta v$ for all v; *i.e.*, $\partial f(0) = [-1, 1]$. Note that this interval contains 0, meaning that the first-order necessary condition (namely, $0 \in \partial f(0)$) is satisfied for 0 to be a minimizer of f, which indeed it is.

The next example, adapted from [6] and [31], shows a function that is differentiable at a given point, but not strictly differentiable there.

Example 3.15 Consider the function $f : \Re \to \Re$ defined by $f(x) = x^2 \left[2 + \sin\left(\frac{\pi}{x}\right)\right]$ with $f(0) = 0$. This function, pictured in Figure 3.1, has a global minimizer at $x = 0$ and possesses infinitely many local optima near 0. One can show that f is differentiable everywhere, including at $x = 0$; thus it is also Lipschitz near $x = 0$. However, the derivative is not continuous at 0, and f is not strictly differentiable at 0, since $\partial f(0) = [-\pi, \pi]$. Also, f is not regular at 0, since $f°(x; \pm 1) = \pi > 0 = f'(x; \pm 1)$, the latter because $f'(0) = 0$.

The following example, taken from [6] and [73], shows a function that is strictly differentiable at a given point, but not continuously differentiable there.

Example 3.16 Consider the convex function, $f : \Re \to \Re$, depicted in Figure 3.2, defined by $f(x) = \int_0^x \varphi(u)du$, where

$$\varphi(u) = \begin{cases} u & \text{if } u \leq 0 \\ \frac{1}{1+\kappa} & \text{if } \kappa + 1 > \frac{1}{u} \geq \kappa \in \mathcal{Z}_+ \end{cases}$$

The function f is Lipschitz near $\hat{x} = 0$. It is shown in [73] that f has kinks at $\frac{1}{\kappa}$, $\kappa = 1, 2, \ldots$, with $\partial f(\frac{1}{\kappa}) = [\frac{1}{\kappa+1}, \frac{1}{\kappa}]$. Since 0 is a limit point of the
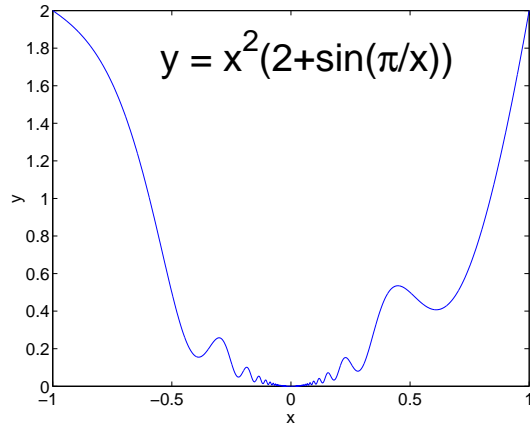
Figure 3.1 A Differentiable Function that is not Strictly Differentiable

sequence $\frac{1}{\kappa}$, f is not strictly differentiable in any neighborhood of $\hat{x} = 0$. The corollary of Proposition 2.2.4 in [31] states that a Lipschitz function f is continuously differentiable in a neighborhood of x if and only if it is strictly differentiable there. Thus, f is not continuously differentiable near $\hat{x}$. Furthermore, $\partial f(0)$ reduces to the singleton $\{0\}$, and the same Proposition ensures that f is strictly differentiable at $\hat{x}$.
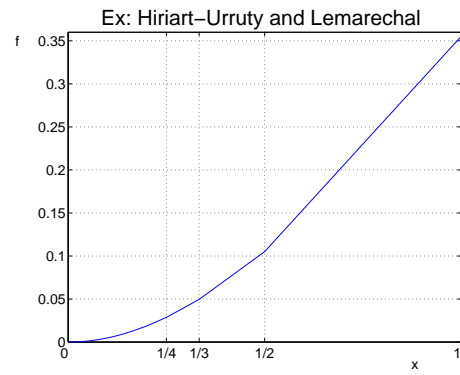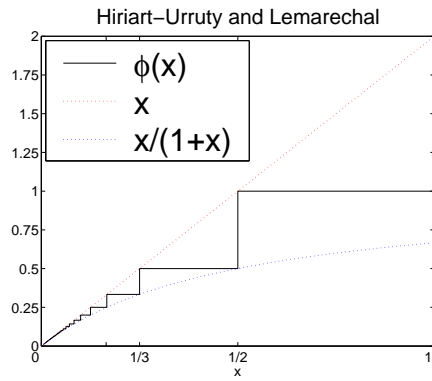
Figure 3.2 A Strictly, but not Continuously Differentiable Function

3.3 New Theoretical Results

The results presented in this section are stated here formally for the first time, although some of the ideas have been shown in proofs by Audet and Dennis [6] in the context of certain limit points of the GPS algorithm. We first define a new concept of D-regularity and relate it to some other concepts from the Clarke calculus. The final theorem will be very useful in later convergence proofs.

>**Definition 3.17** Let D be a set of directions in $\Re^n$. A function f is said to be D-*regular* at $x \in \Re^n$ if, for all $d \in D$, $f'(x;d)$ exists and $f'(x;d) = f^\circ(x;d)$.

Clearly, a function that is regular at a point is also D-regular there for any $D \subset \Re^n$. Lemma 3.18 below provides additional conditions by which a D-regular function can be shown to be regular. This result is then used to relate D-regularity to strict differentiability in Theorem 3.19.

>**Lemma 3.18** Let f be differentiable at x. Then f is regular at x if and only if f is D-regular at x, where D is a positive spanning set.

Proof. Let f be regular at x. Then, by definition, $f'(x;v)$ exists for all $v \in \Re^n$, and f is D-regular for any spanning set D.

Conversely, suppose f is differentiable and D-regular at x, with D a positive spanning set. Let $v \in \Re^n$ be arbitrary. Then $v = \sum_{i=1}^{|D|} \alpha_i d_i$ for some $\alpha_i \geq 0$ and $d_i \in D$, $i = 1, 2, \ldots, |D|$, and by the assumptions and properties of the specified derivatives,

$$
\begin{aligned}
f'(x;v) \quad &\leq \quad f^\circ(x;v) \\
&\leq \quad \sum_{i=1}^{|D|} \alpha_i f^\circ(x;d_i) = \sum_{i=1}^{|D|} \alpha_i f'(x;d_i) = \sum_{i=1}^{|D|} \alpha_i \nabla f(x)^T d_i = \nabla f(x)^T v \\
&= \quad f'(x;v).
\end{aligned}
$$

Thus, f is regular at x.   ∎

Theorem 3.19 Let the function f be strictly differentiable at $x \in \Re^n$. Then f is Lipschitz near x, differentiable at x, and D-regular at x for any set of directions D in $\Re^n$. Conversely, if f is Lipschitz near x, differentiable at x, and D-regular at x, where D is a positive spanning set for $\Re^n$, then f is strictly differentiable at x.

Proof. This follows directly from Theorem 3.13 and Lemma 3.18.   ∎

Example 3.20 shows that the converses of Lemma 3.18 and Theorem 3.19 are not necessarily true if D is not a positive spanning set. That is, it shows a differentiable function that is D-regular, but not regular because D is not a positive spanning set.

Example 3.20 Consider the function $f : \Re^2 \to \Re$, defined by

$$f(x,y) \;=\; \begin{cases} x^2 \left[2 + \sin\left(\frac{\pi}{x}\right) \right], & \text{if } x \neq 0, \\[2mm] 0, & \text{if } x = 0. \end{cases} \tag{3.9}$$

This function has the same properties as that of Example 3.15, except that for $d = (0, \pm 1)$, $f^\circ(0; d) = f'(0; d) = 0$. Thus, as in Example 3.15, f is differentiable (hence Lipschitz) everywhere, but not regular or strictly differentiable at $x = 0$. However, f is D-regular at 0 for $D = \{(0, \pm 1)\}$, which does not positively span $\Re^n$.   ∎

Theorem 3.21 below essentially establishes first-order necessary conditions for optimality with respect to the continuous variables in a mixed variable domain. The assumptions on f given here are slightly weaker than the strict differentiability assumption used in [6] to establish first-order results for GPS limit points – but only in the constrained case. In the unconstrained case, the assumptions on f given here are equivalent to strict differentiability by Theorem 3.19. Although simply assuming

strict differentiability would make the theorem statement shorter, the assumptions made here more clearly convey the relationship between the constrained and uncon- strained cases.

However, we first introduce new notation, so that $f'(x; (d, 0))$ denotes the direc- tional derivative at x with respect to the continuous variables in the direction $d \in \Re^{n^c}$ (*i.e.*, while holding the discrete variables constant – hence the $0 \in \mathcal{Z}^{n^d}$), $f^\circ(x; (d, 0))$ denotes the Clarke generalized directional derivative at x with respect to the contin- uous variables, and $\partial^c f(x)$ represents the generalized gradient of f at x with respect to the continuous variables. This convention is used throughout Chapters 5 and 6 as well.

Theorem 3.21 Let $x = (x^c, x^d) \in X \subseteq \Re^{n^c} \times \mathcal{Z}^{n^d}$. Let the function f satisfy the following conditions:

1. f is Lipschitz near x with respect to the continuous variables,

2. f is differentiable at x with respect to the continuous variables,

3. f is D-regular at x with respect to the continuous variables for a set of vectors D in $\Re^{n^c}$.

If D positively spans the tangent cone $T_{X^c}(x)$, and if $f^\circ(x; (d, 0)) \geq 0$ for all $d \in D \cap T_{X^c}(x)$, then $\nabla^c f(x)^T v \geq 0$ for all $v \in T_{X^c}(x)$. Thus, x is a KKT point of f with respect to the continuous variables. Moreover, if $X^c = \Re^{n^c}$ or if x^c lies in the interior of X^c, then f is strictly differentiable at x with respect to the continuous variables and $0 = \nabla^c f(x) \in \partial^c f(x)$.

Proof. Under the hypotheses given, let D be a set of vectors that positively spans $T_{X^c}(x)$, and let $v \in X^c$ be arbitrary. Then $v = \sum_{i=1}^{|D|} \alpha_i d_i$ for some $\alpha_i \geq 0$ and $d_i \in D$,

$i = 1, 2, \ldots, |D|$. Then

$$\nabla^c f(x)^T v = \sum_{i=1}^{|D|} \alpha_i \nabla^c f(x)^T d_i = \sum_{i=1}^{|D|} \alpha_i f'(x; (d_i, 0)) = \sum_{i=1}^{|D|} \alpha_i f^\circ(x; (d_i, 0)) \geq 0,$$

since all the terms of the final sum are nonnegative.

If $X^c = \Re^{n^c}$, or if x^c lies in the interior of X^c, then $T_{X^c}(x) = \Re^{n^c}$, and D is a positive spanning set; thus, by Corollary 3.19, f is strictly differentiable at x. Furthermore, we have $\nabla^c f(x)^T v \geq 0$ for all $v \in \Re^{n^c}$, and since this holds for $-v$ as well, we also have $\nabla^c f(x)^T v \leq 0$ for all $v \in \Re^{n^c}$. Therefore, $0 = \nabla^c f(x) \in \partial^c f(x)$, (the last step because $\partial^c f(x)$ always contains $\nabla^c f(x)$, if it exists). $\blacksquare$

Chapter 4

GPS for NLP Problems

The purpose of this chapter is to lay out the details of two GPS algorithms for NLP problems. The first section describes the basic GPS algorithm for NLP problems having finitely many linear constraints, while the second details the filter GPS algorithm for NLP problems with general nonlinear constraints. Each section contains a description of the algorithm and a summary of convergence results.

We should note that, since this chapter is meant to clarify already existing algorithms and results, the convergence results remain exactly the same as those found in [6, 7], with the exception of some revised terminology and notation. This is done despite our recent discovery that the key hypothesis of strict differentiability can be slightly weakened in a few of the important theorems without weakening the results. However, the new hypothesis, which is based on that of Theorem 3.21, is used extensively in Chapters 5 and 6 for the more general classes of algorithms (*e.g.*, see Theorems 5.16, 5.17, and 5.20 in the next chapter).

4.1   GPS for Linearly Constrained NLP Problems

In this section, we consider the NLP problem,

$$\min_{x \in X} f(x), \tag{4.1}$$

where $f : X \to \Re \cup \{\infty\}$, $X = \{x \in \Re^n : \ell \le Ax \le u\}$, $A \in \Re^{m \times n}$ is a real matrix, $\ell, u \in (\Re \cup \{\pm\infty\})^n$, and $\ell \le u$.

A key point is that if an iterate falls outside of the domain X, it is simply ignored. This differs in construction from that of [6], [7], and [8], where the algorithm is applied

to $f_X = f + \psi_f$ rather than f, where ψ_f is the indicator function for f; that is, it is zero for any point in X and ∞ outside of X. For the remainder of this document, we use the alternative (but equivalent) construction of ignoring or not considering points outside of X.

We should note also that while we allow $\ell = u$ in the formulation, equality constraints are problematic in practice because points outside of X are not evaluated by the algorithm, and *any* roundoff error would eliminate feasible points from being considered. Thus, to use these methods in practice, variables ought to be eliminated until all the constraints can be expressed as $\ell < u$.

The GPS algorithm is a direction-based method, whose convergence theory is based on searching in directions that form a positive spanning set. A finite set of positive spanning directions D is used to construct (theoretically) a mesh on which GPS iterates lie. This is based on the positive basis theory of Davis [38], and is done with the idea that, if f is sufficiently smooth in a neighborhood of x_k and $\nabla f(x) \neq 0$, then at least one direction $d \in D$ must be a descent direction (see Theorem 3.3). Thus, for some sufficiently small value of $\Delta_k > 0$, there exists a point $y = x_k + \Delta_k d$ satisfying $f(y) < f(x)$. Specifically, at iteration k, the mesh M_k is defined as:

$$M_k = \{x_k + \Delta_k D z \in X : z \in \mathcal{Z}_+^{|D|}\}, \tag{4.2}$$

where x_k is the current iterate, $\Delta_k > 0$ is a parameter that controls the fineness of the mesh, and $\mathcal{Z}_+$ is the set of nonnegative integers. For convenience, the positive spanning set D is represented here as a real $n \times |D|$ matrix whose columns are the vectors in the set.

The positive spanning directions must also satisfy the mild restriction,

$$D = GZ, \tag{4.3}$$

where $G \in \Re^{n \times n}$ is a nonsingular generating matrix, and $Z \in \mathcal{Z}^{n \times |D|}$. A common choice for G is the identity matrix. This construct is essential for the convergence theory and will be discussed in greater detail in Chapter 5. An example of directions not constructed in this manner is provided in [6], in which convergence cannot be guaranteed. Coope and Price [35] use a less restrictive construct in their grid-based methods, but they require that the mesh size be explicitly forced to converge to zero (*i.e.*, it is a hypothesis in their convergence theory).

4.1.1 The Basic GPS Algorithm

The GPS algorithm is formally stated in Figure 4.1. Each iteration is characterized by an optional global SEARCH step and a local POLL step. In the SEARCH step, the objective function f is evaluated at a finite number of points lying on the current mesh M_k in an attempt to try to find a new point with a better function value than the incumbent. Any strategy may be used (including none), as long as the number of mesh points it evaluates is finite. The user can apply a favorite heuristic, such as those described in Section 2.2, or perhaps optimize an inexpensive surrogate function, as is common in difficult engineering design problems with expensive function evaluations [5, 19, 20, 21].

In the POLL step, a positive spanning set $D_k \subseteq D$ is chosen from which to construct the *poll set*. Again, we represent D_k also as a matrix whose columns are the members of the set. It is a function of k and x_k; *i.e.*, $D_k = D(k, x_k) \subseteq D$. The poll set P_k is constructed as the neighboring mesh points in each of the directions in D_k; *i.e.*,

$$P_k = \{x_k + \Delta_k d \in X : d \in D_k\} \tag{4.4}$$

The function f is evaluated at points in P_k until the points have all been evaluated, or until one with a lower objective function value is found.

The set of trial points is defined as $T_k = S_k \cup P_k$, where S_k is the finite set of mesh points evaluated during the SEARCH step. The following definitions define the two possible outcomes of the SEARCH and POLL steps.

Definition 4.1 If $f(y) < f(x_k)$ for some $y \in T_k$, then y is said to be an *improved mesh point.*

Definition 4.2 If $f(x_k) \leq f(y)$ for all $y \in P_k$, then x_k is said to be a *mesh local optimizer.*

If either the SEARCH or POLL step is successful at finding an improved mesh point, then it becomes the new incumbent x_{k+1}, and the mesh is coarsened according to the rule,

$$\Delta_{k+1} = \tau^{m_k^+} \Delta_k, \tag{4.5}$$

where $\tau > 1$ is rational and fixed over all iterations, and the integer m_k^+ satisfies $0 \leq m_k^+ \leq m_{\max}$ for some fixed integer $m_{\max} \geq 0$. Coarsening of the mesh is designed to allow the algorithm to possibly skip over certain local minima and find a better one. It does not prevent convergence of the algorithm, and often makes it faster. Note that only a simple decrease in the objective function value is needed, as is proved in [130].

If the SEARCH and POLL steps both fail to find an improved mesh point, then the incumbent is a mesh local optimizer and remains unchanged (or, alternatively, can be chosen as a point having the same function value as the incumbent, if one exists), while the mesh is refined according to the rule,

$$\Delta_{k+1} = \tau^{m_k^-} \Delta_k, \tag{4.6}$$

where $\tau > 1$ is defined above, $\tau^{m_k^-} \in (0, 1)$, and the integer m_k^- satisfies $m_{\min} \leq m_k^- \leq -1$ for some fixed integer $m_{\min}$.

It follows that, for any integer $k \geq 0$, there exists an integer r_k such that

$$\Delta_k = \tau^{r_k}\Delta_0. \qquad (4.7)$$

Generalized Pattern Search Algorithm – GPS

Initialization: Let x_0 be such that $f(x_0)$ is finite, and let $M_0 \subset X$ be the mesh defined by $\Delta_0 > 0$ and x_0 (see (4.2)).

For $k = 0, 1, 2, \ldots$, perform the following:

1. SEARCH **step:** Employ some finite strategy seeking an improved mesh point; i.e., $x_{k+1} \in M_k$ such that $f(x_{k+1}) < f(x_k)$.

2. POLL **step:** If the SEARCH step was unsuccessful, evaluate f at points in the poll set P_k until an improved mesh point x_{k+1} is found (or until done).

3. **Update:** If SEARCH or POLL finds an improved mesh point,
   Update x_{k+1}, and set $\Delta_{k+1} \geq \Delta_k$ according to (4.5);
   Otherwise, set $x_{k+1} = x_k$, and set $\Delta_{k+1} < \Delta_k$ according to (4.6).

Figure 4.1 Basic GPS Algorithm

To handle bound and linear constraints and still guarantee the convergence results of the basic algorithm, the only requirement is that the directions in D be sufficiently rich to ensure that polling directions can be chosen that conform to the geometry of the constraint boundaries, and that these directions be used infinitely many times. Figure 4.2 depicts a set of conforming directions in $\Re^2$, similar to a drawing in [95]. Audet and Dennis [6] abstract this notion of conformity in Definition 4.3, but appeal to the construction of Lewis and Torczon [95], who actually provide an algorithm for choosing conforming directions using standard linear algebra tools. A slightly restructured but equivalent version of this algorithm is given in Figure 4.3. Note that this algorithm will not work if any of active constraints are degenerate.
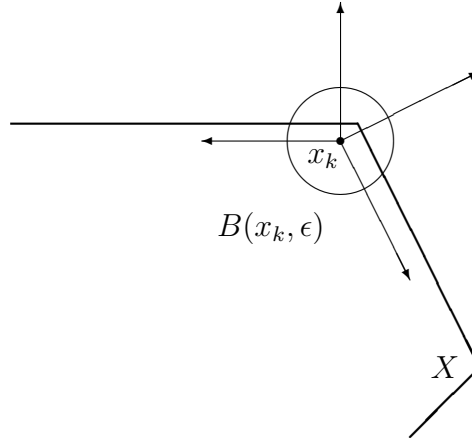
Figure 4.2   Directions that Conform to the Geometry of X

Definition 4.3   A rule for selecting the positive spanning sets $D_k = D(k, x_k) \subseteq D$ *conforms to* X for some $\epsilon > 0$, if at each iteration k and for each y in the boundary of X for which $\|y - x_k\| < \epsilon$, the tangent cone $T_X(y)$ is generated by nonnegative linear combinations of a subset of the columns of D_k.

4.1.2   Summary of Convergence Results

Convergence results for the basic GPS algorithm are presented in detail in [8]. They are only summarized here, since the full theoretical results are presented for more general cases of the algorithm in Chapters 5 and 6.

Algorithm for Computing Conforming Directions D_k

- Set $\epsilon_k \geq \epsilon > 0$. Assume the current iterate x_k satisfies $\ell \leq Ax_k \leq u$.

- While $\epsilon_k \geq \epsilon$, do the following:

  1. Let $I_\ell(x_k, \epsilon_k) = \{i : Ax - \ell \leq \epsilon_k\}$
  2. Let $I_u(x_k, \epsilon_k) = \{i : u - Ax \leq \epsilon_k\}$
  3. Set V denote the matrix whose columns are formed by all the members of the set $\{-a_i : i \in I_\ell(x_k, \epsilon_k)\} \cup \{a_i : i \in I_u(x_k, \epsilon_k)\}$, where a_i^T denotes the i-th row of A.
  4. If V does not have full column rank, then reduce ϵ_k just until $|I_\ell(x_k, \epsilon_k)| + |I_u(x_k, \epsilon_k)|$ is decreased, and return to step 1.

- Set $B = V(V^TV)V^{-1}$ and $N = I - V(V^TV)V^{-1}V^T$.

- Set $D_k = [N, -N, \ B, -B]$.

Figure 4.3 Algorithm for Generating
Conforming Directions (Lewis and Torczon)

To ensure convergence, the following mild assumptions are made:

A1: All iterates $\{x_k\}$ produced by the algorithm lie in a compact set.

A2: The set of directions $D = GZ$, as defined in (4.3), includes tangent cone generators for every point in X.

A3: The rule for selecting positive spanning sets D_k conforms to X for some $\epsilon > 0$.

Assumption A1 already is sufficient to guarantee that there are convergent subsequences of the iteration sequence. However, this is, in fact, a standard assumption [6, 7, 8, 32, 35, 47, 94, 95, 130]. A sufficient condition for this to hold is that the level set $L(x_0) = \{x \in X : f(x) \leq f(x_0)\}$ is compact. We can assume that $L(x_0)$ is bounded, but not closed, since we allow f to be discontinuous and extended valued. Thus we can assume that the closure of $L(x_0)$ is compact. We should also note that

most real engineering optimization problems have simple bounds on the design variables, which is enough to ensure that Assumption A1 is satisfied, since iterates lying outside of X are not evaluated. In the unconstrained case, note that Assumptions A2 and A3 is automatically satisfied by any positive spanning set.

Assumption A2 is automatically satisfied if $G = I$ and the constraint matrix A is rational, as is the case in [95]. Note that the finite number of linear constraints ensures that the set of tangent cone generators for all points in X is finite, which ensure that the finiteness of D is not violated. However, this is not the case for nonlinear constraints, which is the subject of Section 4.2.

If f is lower semi-continuous at any GPS limit point $\bar{x}$, then $f(\bar{x}) \leq \lim_k f(x_k)$, with equality if f is continuous. Of particular interest are limit points of subsequences for which $\lim_k \Delta_k = 0$, which we know exist because of Torczon's [130] key result that $\liminf_k \Delta_k = 0$. The Torczon result is discussed in great detail in Chapter 5. This leads to the following definitions that will be used throughout the remainder of this document.

> **Definition 4.4** A subsequence of GPS mesh local optimizers $\{x_k\}_{k \in K}$ (for some subset of indices K) is said to be a *refining subsequence* if $\{\Delta_k\}_{k \in K}$ converges to zero.

> **Definition 4.5** Let $\hat{x}$ be a limit point of a refining subsequence $\{x_k\}_{k \in K}$. A direction $d \in D$ is said to be a *limit direction* of $\hat{x}$ if $w_k = x_k + \Delta_k d \in X$ and $f(x_k) \leq f(w_k)$ for infinitely many $k \in K$.

Audet and Dennis [8] prove the existence of at least one convergent refining subsequence. An important point is that, since a limit direction d is one in which $x_k + \Delta_k d \in X$ infinitely often, it must be a feasible direction at the $\hat{x}$, and thus lies in the tangent cone $T_X(\hat{x})$.

The key results of Audet and Dennis are now given. The first shows directional optimality conditions under the assumption of Lipschitz continuity, and is obtained by a very short and elegant proof using Clarke's [31] definition of the generalized directional derivative. Following this result is an example showing a very simple convex function, in which the directional optimality conditions hold, but no stronger result can be obtained. The final two results in this section, which follow from Theorem 3.21, show convergence to points satisfying first-order necessary optimality conditions under the assumption of strict differentiability.

Theorem 4.6 Let $\hat{x}$ be a limit of a refining subsequence, and let $d \in D$ be any limit direction of $\hat{x}$. Under Assumptions A1–A3, if f is Lipschitz near $\hat{x}$, then the generalized directional derivative of f at $\hat{x}$ in the direction d is nonnegative, *i.e.*, $f^\circ(\hat{x}; d) \geq 0$.

Example 4.7 Let $f : \Re^2 \to \Re$ be defined by $f(x) = \|x\|_\infty$. Let $x_0 = (1, 1)$, and $D = \{\pm e_1, \pm e_2\}$, and choose an empty SEARCH step. For *any*
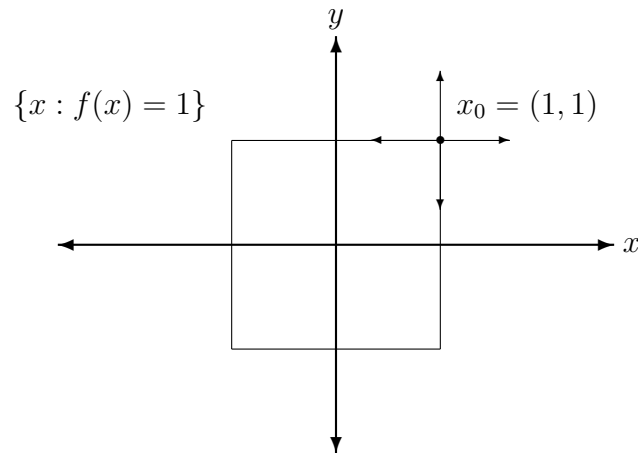
Figure 4.4 An Illustrative Example of Directional Optimality

value of $\Delta > 0$ and for all $d \in D$, it is not difficult to see that $x_0 + \Delta d$

cannot yield an improved mesh point. (The directions e_1 and e_2 are ascent directions, while the directions $-e_1$ and $-e_2$ are parallel to the level sets of f.) Thus, the algorithm will stall at $\hat{x} = (1, 1)$ without moving.

It is easy to show that f is Lipschitz everywhere, but not differentiable at $\hat{x} = (1, 1)$. Furthermore, it is clear that Theorem 4.6 holds; *i.e.*, $f^\circ(\hat{x}; d) \geq 0$ for each $d \in D$. However, any direction that lies in the interior of the cone generated by $-e_1$ and $-e_2$ is a descent direction (none of these directions belong to D).

Note that the specific choices of x_0 and D force this unusual behavior, which can be remedied by incorporating either a slight perturbation in the starting point or poll directions, or any reasonable nonempty SEARCH step. ∎

Theorem 4.8 Under Assumptions A1–A3, if f is strictly differentiable at a limit point $\hat{x}$ of a refining subsequence, then $\nabla f(\hat{x})^T w \geq 0$ for all $w \in T_X(\hat{x})$, and $-\nabla f(\hat{x}) \in N_X(\hat{x})$. Thus, $\hat{x}$ satisfies the KKT first-order necessary conditions for optimality.

Corollary 4.9 Under Assumption A1, let $X = \Re^n$ and $\hat{x}$ be any limit of a refining subsequence. If f is strictly differentiable at $\hat{x}$, then $\nabla f(\hat{x}) = 0$.

4.2 GPS for General Constrained NLP Problems

Nonlinear constraints pose a problem for the basic GPS algorithm in that choosing enough directions to conform to the geometry of the constraints (to guarantee convergence to a KKT point) would require an infinite number of directions in D, which the convergence theory does not support. Thus, for NLP problems with nonlinear constraints, a different strategy must be employed.

To address nonlinear constraints, Lewis and Torczon [96] introduce a derivative-free augmented Lagrangian pattern search method, based on the approach developed by Conn, Gould, and Toint [32], for solving NLP problems if the functions are continuously differentiable. They are able to prove convergence to a KKT point, under the assumption of continuous differentiability of the objective and constraint functions on the domain of the problem. However, the method requires estimates of Lagrange multipliers and a penalty parameter that must be updated at every iteration. Currently, no numerical performance results have been published.

As an alternative, Fletcher and Leyffer [47] developed the filter algorithm as a way to globalize SQP and SLP methods without the use a merit or penalty function, which would require the user to specify the relative weighting of optimality versus feasibility. In filter algorithms, the goal is to minimize two functions, the objective f and a continuous aggregate constraint violation function h that satisfies $h(x) \geq 0$ with $h(x) = 0$ if and only if x is feasible. The function h is often set to $h(x) = \|C(x)_+\|$, where $\|\cdot\|$ is a vector norm and $C(x)_+$ is the vector of constraint violations at x; i.e., for $i = 1, 2, \ldots, m$, $C_i(x)_+ = C_i(x)$ if $C_i(x) > 0$; otherwise, $C_i(x)_+ = 0$. In particular, if the squared 2-norm is used, then h inherits whatever smoothness properties C possesses [7].

4.2.1 The Filter GPS Algorithm

The concept of a filter is based on a notion of dominance. The definition of dominance provided below, which comes from the multi-criteria optimization literature, is adapted from a similar term in [47], so that it is defined with respect to the objective function f and constraint violation function h. A formal definition of a filter follows immediately thereafter.

Definition 4.10 A point $x \in \Re^n$ is said to *dominate* $y \in \Re^n$, written $x \prec y$, if $f(x) \leq f(y)$ and $h(x) \leq h(y)$ with either $f(x) < f(y)$ or $h(x) < h(y)$

Definition 4.11 A *filter*, denoted $\mathcal{F}$, is a finite set of points such that no pair of points x and y in the set have the relation $x \prec y$.

In constructing a filter for GPS, we put two additional restrictions on $\mathcal{F}$. First, we set a bound on aggregate constraint violation, $h_{\max}$ so that each point $y \in \mathcal{F}$ satisfies $h(y) < h_{\max}$. Second, we include only infeasible points in the filter and track feasible points separately. This is done in order to avoid a problem with what Fletcher and Leyffer [47] refer to as "blocking entries", in which a feasible filter point with lower function value than a nearby local minimum prevents convergence to both that minimum and a global minimum. Tracking feasible points outside of the filter circumvents this uncommon but plausible scenario. With these two modifications, the following terminology is now provided.

Definition 4.12 A point x is said to be *filtered* by a filter $\mathcal{F}$ if any of the following properties hold:

1. There exists a point $y \in \mathcal{F}$ such that $y \preceq x$,

2. $h(x) \geq h_{\max}$,

3. $h(x) = 0$ and $f(x) \geq f^F$, where f^F is the objective function value of the best feasible point found thus far.

The point x is said to be *unfiltered* by $\mathcal{F}$ if it is not filtered by $\mathcal{F}$.

Thus, the set of unfiltered points, denoted by $\overline{\mathcal{F}}$, is given by

$$\overline{\mathcal{F}} = \bigcup_{x \in \mathcal{F}} \{y : y \succeq x\} \cup \{y : h(y) \geq h_{\max}\} \cup \{y : h(y) = 0, f(y) \geq f^F\}. \qquad (4.8)$$

Observe that, with this notation, if a new trial point has the same function values as those of any point in the filter, then the trial point is filtered. Thus, only the first point with such values is accepted into the filter.

In unconstrained GPS, the POLL step is performed by evaluating the objective function at the mesh neighbors of the current iterate (*i.e.*, the poll set) until an improvement is found. When the filter is added for general NLP problems, the poll set consists of the mesh neighbors of the poll center p_k, which is chosen either as the incumbent best feasible point p_k^F or the incumbent least infeasible point p_k^I (I and F denote infeasible and feasible, respectively). The poll set is centered at p_k therefore defined as

$$P_k(p_k) = \{p_k + \Delta_k d \in X : d \in D_k\}, \tag{4.9}$$

where $D_k \subset D$ is the set of poll directions at iteration k. For convenience, we use the notation, $h_k^I = h(p_k^I) > 0$, $f_k^I = f(p_k^I)$, and $f_k^F = f(p_k^F)$. The flexibility in selecting either point as the poll center does not affect convergence behavior, other than converging to a different limit point [7]. The bi-loss graph in Figure 4.5 depicts a filter with its possible poll centers for the next iteration. Observe that feasible points lie on the vertical axis (labelled f).

We should note that polling around other points in the filter is certainly allowed; however, other points do not possess the same convergence properties as p_k^F and p_k^I in the limit. Therefore, any such polling should be regarded as part of the SEARCH step, so that the convergence theory is preserved.

The next two definitions describe the two possible outcomes of the SEARCH and POLL steps.

The figure shows axes with f on the vertical axis and h on the horizontal axis. Points f_k^F and (h_k^I, f_k^I) are marked, with a stepped region labeled $\overline{\mathcal{F}}_k$ and $h_{\max}$ marked on the horizontal axis.

Feasible region:	$\{x \in \Re^n : h(x) = 0\}$
Trial set:	T_k
Filtered points:	$\overline{\mathcal{F}}_k$
Unfiltered points found:	$T_k \setminus \overline{\mathcal{F}}_k \neq \emptyset$
Mesh isolated filter points:	$T_k \subset \overline{\mathcal{F}}_k$
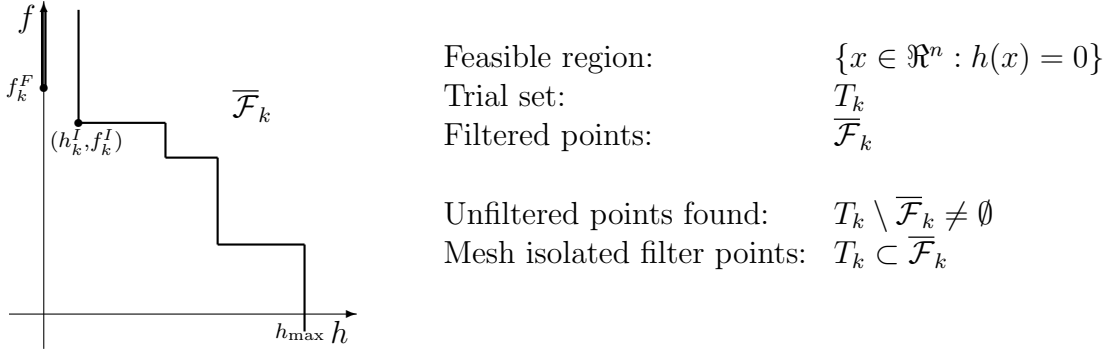
Figure 4.5   Graphical Depiction of a Filter and Types of Iterations.

Definition 4.13   Let T_k denote the set of trial points to be evaluated at iteration k, and let $\overline{\mathcal{F}}_k$ denote the set of filtered points described by (4.8). A point $y \in T_k$ is said to be an *unfiltered point* if $y \notin \overline{\mathcal{F}}_k$.

Definition 4.14   Let $P_k(p_k)$ denote the poll set centered at the point p_k, and let $\overline{\mathcal{F}}_k$ denote the set of filtered points described by (4.8). The point p_k is said to be a *mesh isolated filter point* if $P_k(p_k) \subset \overline{\mathcal{F}}_k$. If $p_k \in \{p_k^F, p_k^I\}$, then p_k is said to be a *mesh isolated poll center.*

At iteration k, if an unfiltered point is found in either the SEARCH or POLL step, then the mesh is coarsened according to the rule in (4.5), and the poll center is updated, if appropriate. If an unfiltered point is not found, then the poll center p_k is a mesh isolated filter point, and the mesh is refined according to the rule in (4.6).

The Filter GPS algorithm for NLPs is presented in Figure 4.6. The algorithm requires an initial population of the filter, which can be done in any manner stated in [7]. Note that, if there are no constraints, then $h = 0$, and an unfiltered point only applies to the single function value of f. Thus, in this case, the Filter GPS algorithm reduces to the basic version shown in Figure 4.1.

Generalized Filter Pattern Search Algorithm – FGPS

Initialization: Let x_0 be an undominated point of a set of initial solutions. Include all these points in the filter $\mathcal{F}_0$, together with $h_{\max} > h(x_0)$. Fix $\Delta_0 > 0$.

For $k = 0, 1, 2, \ldots$, perform the following:

1. Update poll center $p_k \in \{p_k^F, p_k^I\}$.

2. Compute incumbent values $f_k^F = f(p_k^F), h_k^I = h(p_k^I), f_k^I = f(p_k^I)$.

3. SEARCH **step:** Employ some finite strategy seeking an unfiltered mesh point $x_{k+1} \notin \overline{\mathcal{F}}_k$.

4. POLL **step:** If the SEARCH step did not find an unfiltered point, evaluate f and h at points in the poll set $P_k(p_k)$ until an unfiltered mesh point $x_{k+1} \notin \overline{\mathcal{F}}_k$ is found (or until done).

5. **Update:** If SEARCH or POLL found an unfiltered mesh point,
   Update filter $\mathcal{F}_{k+1}$ with x_{k+1}, and set $\Delta_{k+1} \geq \Delta_k$ according to (4.5);
   Otherwise, set $\mathcal{F}_{k+1} = \mathcal{F}_k$, and set $\Delta_{k+1} < \Delta_k$ according to (4.6).

Figure 4.6 Filter GPS Algorithm

4.2.2 Summary of Convergence Results

Convergence results for the filter GPS algorithm are presented in detail in [7]. They are only summarized here, since the full theoretical results are presented for the more general case of mixed variable problems in Chapter 6. The same assumptions as those listed in Section 4.1.2 apply.

Mesh Size Behavior

The behavior of Δ_k is almost identical to that of Δ_k for the basic GPS algorithm. Because the same assumptions hold (in particular, all iterates lie in a compact set),

the results of Torczon [130] apply; namely, that Δ_k is bounded above by a positive constant independent of k, and $\liminf_{k \to +\infty} \Delta_k = 0$.

The addition of the filter makes it necessary to redefine the ideas of a refining subsequence and a limit direction; however, the definitions now provided generalize the previous ones, in that they are equivalent to the those of the basic GPS method if there are no nonlinear constraints.

Definition 4.15 A subsequence of mesh isolated poll centers $\{p_k\}_{k \in K}$ (for some subset of indices K) is said to be a *refining subsequence* if $\{\Delta_k\}_{k \in K}$ converges to zero.

Definition 4.16 Let $\hat{p}$ is a limit point of a refining subsequence $\{p_k\}_{k \in K}$. A direction $d \in D$ is said to be a *limit direction* of $\hat{p}$ if $p_k + \Delta_k d$ belongs to X and is filtered for infinitely many $k \in K$.

In [7] Audet and Dennis prove the existence of convergent refining subsequences and show that any such subsequence has at least one limit point and one positive spanning set of limit directions.

Results for the Constraint Violation Function

For the constraint violation function h, the following two properties hold without assuming any constraint qualification.

Theorem 4.17 Let $\hat{p}$ be a limit point of a refining subsequence. Under Assumptions A1–A3, if h is Lipschitz continuous near $\hat{p}$, then $h^\circ(\hat{p}; d) \geq 0$ for all limit directions $d \in D$ of $\hat{p}$.

Theorem 4.18 Under Assumption A1, let $\hat{p}$ be a limit of a refining subsequence. If h is strictly differentiable at $\hat{p}$, then $\nabla h(\hat{p}) = 0$.

Although we would like to converge to a point $\hat{p}$ that is feasible (*i.e.*, $h(\hat{p}) = 0$), Theorem 4.18 only guarantees convergence to a first-order stationary point of h. Still, any feasible point is also a stationary point – in fact, a global minimizer of h; therefore, the algorithm can (and often will) converge to such a feasible point. If $\hat{p}$ is not feasible, then $\hat{p}$ still satisfies first-order conditions for being a local minimizer of h. This result may seem less advantageous, but it is no different than for many gradient-based algorithms, such as sequential quadratic programming [114].

Results for the Objective Function

The following results apply to the objective function f. Note that, in general, convergence to a KKT point is not guaranteed, in that the negative gradient does not necessarily lie inside the normal cone at the limit point. However, Theorem 4.20 specifies a cone contained in the tangent cone at the limit point, whose polar contains the negative gradient.

> **Theorem 4.19** Let $\hat{p}$ be a limit point of a refining subsequence $\{p_k\}_{k \in K}$, and let $d \in D$ be a limit direction of $\hat{p}$. Under Assumptions A1–A3, if f is Lipschitz continuous near $\hat{p}$ and $f(p_k) \leq f(p_k + \Delta_k d)$ for infinitely many k in K, then $f^\circ(\hat{p}; d) \geq 0$.

> **Theorem 4.20** Let $\hat{p}$ be a limit point of a refining subsequence $\{p_k\}_{k \in K}$, and let V_d be the cone generated by all limit directions $d \in D$ of $\hat{p}$, for which $f(p_k) \leq f(p_k + \Delta_k d)$ infinitely often. Under Assumptions A1–A3, if f is strictly differentiable at $\hat{p}$, then $-\nabla f(\hat{p})$ belongs to the polar V_d° of V_d.

Other corollaries associated with the Filter GPS algorithm assume that $C(\hat{x}) < 0$ holds, in which case, the results are identical to those of Section 4.1. Thus, this algorithm is indeed a generalization of the basic GPS algorithm.

One other key result must be mentioned. The following theorem shows that if f is sufficiently smooth, the algorithm will not stall at any non-stationary point, regardless of the positive spanning set chosen.

> **Theorem 4.21** If h and f are strictly differentiable at poll center p_k, and if $\nabla f(p_k) \neq 0$, then there cannot be infinitely many consecutive iterations where p_k is a mesh isolated filter point.

Chapter 5

GPS for Linearly Constrained MVP Problems

In this chapter, the class of Mixed Variable Generalized Pattern Search (MGPS) algorithms is described. The algorithm, which is developed and presented first, is the work of Audet and Dennis [6]. Some notation is changed slightly, and some additional definitions are provided for the sake of clarity. However, the convergence results that follow are new and apply to a broader class of problems, as the smoothness assumptions of Audet and Dennis have been relaxed. This algorithm is applied successfully in [89] to the design of a thermal insulation system, an expanded version of which is the subject of Chapter 7.

We remind the reader that, as discussed in Chapter 1, categorical variables are those that must take on values from a pre-defined list (*e.g.*, color, material type). They are often assigned numeric values, but such values may be meaningless. The key point is that these variables cannot be relaxed so as to take on other values, as would be done in most algorithms for solving mixed integer problems. The addition of categorical variables can also create other challenges not common in most optimization problems. In particular, a change in discrete variable values can generate a change in the constraints, and even a change in the dimension of the problem.

For mixed variable problems, we partition each point x into its continuous and discrete parts x^c and x^d, respectively; *i.e.*, $x = (x^c, x^d)$, where $x^d \in \mathcal{Z}^{n^d}$ and $x^c \in \Re^{n^c}$, and where n^c and n^d denote the maximum dimensions of the continuous and discrete variables, respectively. (By convention, unused variables are ignored.)

The problem under consideration in this chapter is then expressed as follows:

$$\min_{x \in X} f(x) \tag{5.1}$$

where $f : X \subset \Re^{n^c} \times \mathcal{Z}^{n^d} \to \Re \cup \{\infty\}$, where X is partitioned into respective continuous and discrete variable spaces, $X^c \subseteq \Re^{n^c}$ and $X^d \subseteq \mathcal{Z}^{n^d}$, with respective dimensions n^c and n^d. Furthermore, the continuous variable space is defined by a finite set of linear constraints, dependent on the values of x^d; *i.e.*,

$$X^c(x^d) = \{x^c \in \Re^{n^c} : \ell(x^d) \le A(x^d)x^c \le u(x^d)\}, \tag{5.2}$$

where $A(x^d) \in \Re^{m^c \times n^c}$ is a real matrix, $\ell(x^d), u(x^d) \in (\Re \cup \{\pm\infty\})^{n^c}$, and $\ell(x^d) \le u(x^d)$ for all values of x^d.

As noted in the previous chapter, equality constraints are problematic in practice because points outside of X are not evaluated by the algorithm, and *any* roundoff error would eliminate feasible points from being considered. Thus, to use these methods in practice, variables ought to be eliminated until all the constraints can be expressed as $\ell(x^d) < u(x^d)$ for all values of x^d.

If $n^d = 0$, then the problem reduces to a standard linearly constrained NLP problem, in which ℓ, A, and u do not change. For convenience, we now omit the explicit dependence on x^d in this chapter, even though it is understood for MVP problems.

5.1 Mesh Construction and Local Optimality

The MGPS algorithm is an extension of the basic GPS algorithm in which discrete or categorical variables are permitted. To accommodate this change, the mesh must be defined differently, but in a way that reduces to that of the basic GPS mesh structure if there are no discrete variables. The construction here is slightly more general here than in [8], in that no single generating matrix is required.

For each combination $i = 1, 2, \ldots, i_{\max}$, of values that the discrete variables may take, a set of positive spanning directions D^i is constructed by forming the product

$$D^i = G_i Z_i, \tag{5.3}$$

where $G_i \in \Re^{n^c \times n^c}$ is a nonsingular generating matrix, and $Z_i \in \mathcal{Z}^{n^c \times |D^i|}$. We will sometimes use $D(x)$ in place D^i to indicate that the set of directions is associated with the discrete variable values of $x \in X$. The set D is then defined by $D = \bigcup_{i=1}^{i_{\max}} D^i$.

The mesh is formed as the direct product of X^d with the union of a finite number of lattices in X^c; *i.e.*,

$$M_k = X^d \times \bigcup_{i=1}^{i_{\max}} \{x_k^c + \Delta_k D^i z \in X^c : z \in \mathcal{Z}_+^{|D^i|}\}. \tag{5.4}$$

We should note that the mesh is purely conceptual and is never explicitly created. Instead, directions are only generated when necessary in the algorithm.

In order to solve problems with categorical variables, a notion of local optimality is needed. For continuous variables, this is well-defined in terms of local neighborhoods. However, for categorical variables, a local neighborhood must be defined by the user, and there may be no obvious choice for doing so; special knowledge of the underlying engineering process or physical problem may be the only guide.

To make the definition as general as possible, we define local neighborhoods in terms of a set-valued function $\mathcal{N} : X \to 2^X$, where 2^X denotes the power set (or set of all possible subsets of X). By convention, we assume that for all $x \in X$, the set $\mathcal{N}(x)$ is finite, and $x \in \mathcal{N}(x)$.

For our analysis, we also require a definition of a limit in the mixed variable domain, along with a notion of continuity for $\mathcal{N}$, given by the following definitions:

Definition 5.1 Let $X \subseteq (\Re^{n^c} \times \mathcal{Z}^{n^d})$ be a mixed variable domain. A sequence $\{x_i\} \subset X$ is said to *converge* to $x \in X$ if, for every $\epsilon > 0$, there

exists a positive integer N such that $x_i^d = x^d$ and $\|x_i^c - x^c\| < \epsilon$ for all $i > N$. The point x is said to be the *limit point* of the sequence $\{x_i\}$.

Definition 5.2 The set-valued function $\mathcal{N} : X \subseteq (\Re^{n^c} \times \mathcal{Z}^{n^d}) \to 2^X$ is said to be *continuous* at $x \in X$ if, for every $\epsilon > 0$, there exists $\delta > 0$ such that, whenever $u \in X$ satisfies $u^d = x^d$ and $\|u^c - x^c\| < \delta$, the following two conditions hold:

1. If $y \in \mathcal{N}(x)$, then there exists $v \in \mathcal{N}(u)$ satisfying $v^d = y^d$ and $\|v^c - y^c\| < \epsilon$,

2. If $v \in \mathcal{N}(u)$, then there exists $y \in \mathcal{N}(x)$ satisfying $y^d = v^d$ and $\|y^c - v^c\| < \epsilon$.

Definition 5.2 will ensure in Theorem 5.12 that, given a certain convergent subsequence of iterates and a corresponding convergent subsequence of points that are discrete neighbors of these iterates, the limit point of the discrete neighbor points is itself a discrete neighbor of the limit point of the iterates.

As an example, one common choice of neighborhood function for integer variables is the one defined by $\mathcal{N}(x) = \{y \in X^d : \|y - x\|_1 \leq 1\}$. However, categorical variables may have no inherent ordering, which would make this choice inapplicable.

With a well-defined neighborhood function, we can now extend the classical definition of local optimality to mixed variable domains, by the following slight modification of a similar definition by Audet and Dennis [8].

Definition 5.3 A point $x = (x^c, x^d) \in X$ is said to be a *local minimizer of f with respect to the set of neighbors $\mathcal{N}(x) \subset X$* if there exists an $\epsilon > 0$ such that $f(x) \leq f(v)$ for all v in the set

$$X \cap \bigcup_{y \in \mathcal{N}(x)} \left(B(y^c, \epsilon) \times y^d \right). \tag{5.5}$$

The function $\mathcal{N}$ also must be constructed so that every discrete neighbor of the current iterate lie on the mesh; *i.e.*, $\mathcal{N} \subseteq M_k$ for all $k = 0, 1, \ldots$. This will be explicitly stated as an assumption in Section 5.3. Also observe the each lattice in (5.4) is expressed as a translation from x_k^c, as opposed to y_k^c, for some $y_k \in \mathcal{N}(x_k)$. This is necessary to ensure convergence of the algorithm.

Note that this construction does not require that a point and its discrete neighbors have the same continuous variable values; rather, it requires that they both lie on the same mesh. In fact, Kokkolaras *et al.* [89] construct their neighbor sets in a way that neighbors often do *not* have the same continuous variable values.

5.2 The MGPS Algorithm

Just like the basic GPS algorithm, each iteration of the MGPS algorithm consists of a SEARCH and possibly a POLL step. The SEARCH step is simply a global search of a finite number of trial points on the mesh. It is almost identical to that of the basic GPS algorithm, and is discussed in detail in Chapters 2 and 4.

Polling in the MGPS algorithm is performed with respect to the continuous variables, the discrete neighbor points, and the set of points generated by an EXTENDED POLL step.

At iteration k, let $D_k^i \subseteq D^i$ denote the set of poll directions corresponding to the i-th set of discrete variable values, and we define $D_k = \bigcup_{i=1}^{i_{\max}} D_k^i$. Again, we will often write D_k^i as $D_k(x_k)$ to indicate that the polling directions are associated with the discrete variable values of x_k. The *poll set* centered at x_k is defined as

$$P_k(x_k) = \{x_k + \Delta_k(d, 0) \in X : d \in D_k(x_k)\}. \tag{5.6}$$

We remind the reader that the notation $(d, 0)$ is consistent with the partitioning into continuous and discrete variables, respectively, where 0 means that discrete variables do not change value. Thus, $x_k + \Delta_k(d, 0) = (x_k^c + \Delta_k d, x_k^d)$.

In some cases where the poll set and set of discrete neighbors fail to produce a lower objective function value, MGPS performs an EXTENDED POLL step, in which additional polling is performed around any points in the set of discrete neighbors whose objective function value is sufficiently close to the incumbent value. That is, if $y \in \mathcal{N}(x_k)$ satisfies $f(x_k) \leq f(y) < f(x_k) + \xi_k$ for some user-specified tolerance value $\xi_k \geq \xi$ (called the *extended poll trigger*), where ξ is a fixed positive scalar, then we begin a polling sequence $\{y_k^j\}_{j=1}^{J_k}$ with respect to the continuous neighbors of y_k beginning with $y_k^0 = y_k$ and ending when either $f(y_k^{J_k} + \Delta_k(d, 0)) < f(x_k)$ for some $d \in D_k(y_k^{J_k})$, or when $f(x_k) \leq f(y_k^{J_k} + \Delta_k(d, 0))$ for all $d \in D_k(y_k^{J_k})$. For this discussion, we let $z_k = y_k^{J_k}$, the last iterate (or *endpoint*) of the EXTENDED POLL step. We should note that in practice, the parameter ξ_k is typically set as a percentage of the objective function value (but bounded away from zero), such as, say, $\xi_k = \max\{\xi, 0.05|f(x_k)|\}$. A relatively high choice of ξ_k will generate more EXTENDED POLL steps, thus making a solution more global, but it will require more function evaluations. On the other hand, a lower value of ξ_k will cost fewer function evaluations, but it will tend to make the solution more local.

The set of extended poll points for a discrete neighbor $y \in \mathcal{N}(x_k)$, denoted $\mathcal{E}(y)$, contains a subset of the points $\{P_k(y_k^j)\}_{j=1}^{J_k}$. At iteration k, the set of points evaluated in the EXTENDED POLL step (or *extended poll set*) is given by

$$\mathcal{X}_k(\xi_k) = \bigcup_{y \in \mathcal{N}_k^\xi} \mathcal{E}(y), \tag{5.7}$$

where $\mathcal{N}_k^\xi = \{y \in \mathcal{N}(x_k) : f(x_k) \leq f(y) \leq f(x_k) + \xi_k\}$.

The set of trial points is defined as $T_k = S_k \cup P_k(x_k) \cup \mathcal{N}(x_k) \cup \mathcal{X}_k(\xi_k)$, where S_k is the finite set of mesh points evaluated during the SEARCH step. Similar to the basic GPS algorithm described in Section 4.1, the following definitions define the two outcomes of the SEARCH, POLL, and EXTENDED POLL steps.

Definition 5.4   If $f(y) < f(x_k)$ for some $y \in T_k$, then y is said to be an *improved mesh point.*

Definition 5.5   If $f(x_k) \leq f(y)$ for all $y \in P_k(x_k) \cup \mathcal{N}(x_k) \cup \mathcal{X}_k(\xi_k)$, then x_k is said to be a *mesh local optimizer.*

These trial points are illustrated in Figure 5.1, where they are indexed by iteration number k. In the lower plane, the sequence of MGPS iterates is given by the points x_k with limit point $\hat{x}$. (We assume that k is sufficiently large, so that $x_k^d = x_{k+1}^d = \ldots = \hat{x}^d$.) In the upper plane, the sequence of discrete neighbor points represented by $y_k \in \mathcal{N}(x_k)$ converges to $\hat{y} \in \mathcal{N}(\hat{x})$. For each iteration k, the EXTENDED POLL step is represented by the sequence $y_k, \ldots, y_k^j, \ldots, z_k$, where the sequence of extended poll endpoints represented by z_k converges to limit point $\hat{z}$, and intermediate extended poll centers y_k^j converge to limit point(s) $\bar{y}$. We should note that existence of the limit points described here has not yet been established, but will be proven in Section 5.3, and in particular, Theorem 5.12.

Figure 5.2 shows an example with one discrete and two continuous variables, in which the set of neighbors of the current iterate x_k is assumed to be $\mathcal{N}(x_k) = \{x_k, y_1^0, y_2^0\}$. (The subscripts 1 and 2 are added to distinguish the points in the set of neighbors $\mathcal{N}(x_k)$. Note that the points in $\mathcal{N}(x_k)$ do not have the same continuous variable values). The iterate x_k is a local minimizer of f on the set $P_k(x_k) \cup \mathcal{N}(x_k) \cup \mathcal{X}_k(\xi_k)$ if, for some $\epsilon > 0$, $f(x_k) \leq f(v)$ for all $v \in B(x_k, \epsilon) \cup B(y_1^0, \epsilon) \cup B(y_2^0, \epsilon)$. The
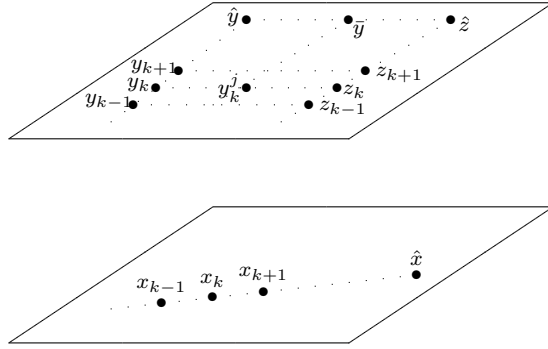
Figure 5.1 Limit Points of Iterates and Extended Poll Centers.

letters a to e, i to l, and u to w in the figure represent poll points around the different poll centers, x_k, y_1^0, y_1^1, and y_2^0.

The MGPS algorithm is presented formally in Figure 5.3. Note that updating of the current iterate and mesh size parameter are done in exactly the same way as in the basic GPS algorithm presented in Figure 4.1.

5.3 New Convergence Results

A key assumption in the original Audet and Dennis GPS paper [8] for mixed variable problems, consistent with the assumptions of Torczon [130] and Lewis and Torczon [94, 95], is that the objective function is continuously differentiable with respect to the continuous variables on a neighborhood of the level set $L_X(x_0) = \{x \in X : f(x) \leq f(x_0)\}$, where $x_0 \in X$ is the initial iterate. This section is devoted to new convergence results for the bound and linearly constrained cases, where assumptions on the objective function are relaxed, so that the theory is consistent with [6] and [7], as described in Chapter 4. Specifically, we use the same tools from the non-smooth calculus of Clarke [31] to provide a more complete characterization of the limit points. This approach is continued in Chapter 6 so that the FMGPS algorithm

$$P_k(x_k) = \{u, v, w\}$$

$$\mathcal{N}(x_k) = \{x_k, y_1^0, y_2^0\}$$

$$f(x_k) \leq f(y_1^0) < f(x_k) + \xi_k < f(y_2^0)$$

$$\mathcal{X}_k(\xi_k) = \{y_1^1\} \cup \{a, b, c\}$$
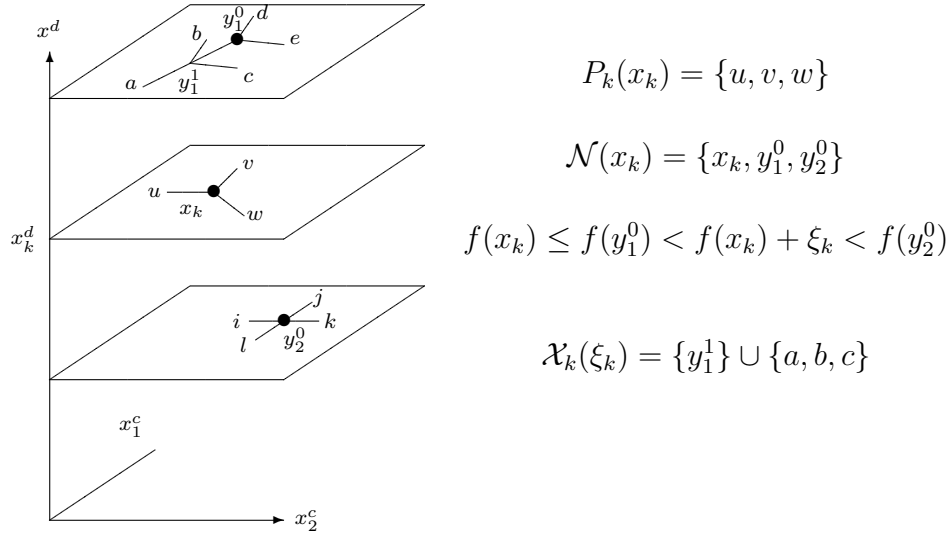
Figure 5.2   Construction of the Poll and Extended Poll Sets

presented there will unify all of the GPS theory to date into one framework for an extremely broad class of optimization problems.

The following assumptions are made, the first three of which are identical to those of Section 4.1.2:

A1: All iterates $\{x_k\}$ produced by the MGPS algorithm lie in a compact set.

A2: For each fixed set of discrete variable values x^d, the corresponding set of directions $D^i = G_i Z_i$, as defined in (5.3), includes tangent cone generators for every point in $X^c(x^d)$.

A3: The rule for selecting directions D_k conforms to X^c for some $\epsilon > 0$ (see Definition 4.3).

A4: The discrete neighbors always lie on the mesh; i.e., $\mathcal{N}(x_k) \subset M_k$ for all k

---

The first result, due to Audet and Dennis [6], describes the behavior of iteration sequences under the mildest assumptions.

General Mixed Variable Pattern Search Algorithm - MGPS

Initialization: Let $x_0 \in X$ satisfy $f(x_0) < \infty$. Set $\Delta_0 > 0$ and $\xi > 0$.

For $k = 0, 1, 2, \ldots$, perform the following:

1. Set extended poll trigger $\xi_k \geq \xi$.

2. SEARCH **step:** Employ some finite strategy seeking an improved mesh point; i.e., $x_{k+1} \in M_k$ such that $f(x_{k+1}) < f(x_k)$.

3. POLL **step:** If the SEARCH step does not find an improved mesh point, evaluate f at points in $P_k(x_k) \cup \mathcal{N}(x_k)$ until an improved mesh point x_{k+1} is found (or until done).

4. EXTENDED POLL **step:** If the SEARCH and POLL steps does not find an improved mesh point, evaluate f at points in $\mathcal{X}_k(\xi_k)$ until an improved mesh point x_{k+1} is found (or until done).

5. **Update:** If SEARCH, POLL, or EXTENDED POLL finds an improved mesh point, Update x_{k+1}, and set $\Delta_{k+1} \geq \Delta_k$ according to (4.5); Otherwise, set $x_{k+1} = x_k$, and set $\Delta_{k+1} < \Delta_k$ according to (4.6).

Figure 5.3 MGPS Algorithm

Theorem 5.6 Under Assumption A1, there exists at least one limit point of the iteration sequence $\{x_k\}$. If f is lower semicontinuous at such a limit point $\bar{x}$ with respect to the continuous variables, then $\lim_k f(x_k)$ exists and is greater than or equal to $f(\bar{x})$, which is finite. If f is continuous at every limit point of $\{x_k\}$, then every limit point has the same function value.

Proof. Since f is lower semicontinuous at $\bar{x}$, we know that for any subsequence $\{x_k\}_{k \in K}$ of the iteration sequence that converges to $\bar{x}$, $\liminf_{k \in K} f(x_k) \geq f(\bar{x})$, which is finite because $f(x_k)$ is finite for each $k \in K$. But the subsequence of function

values is a subsequence of a nonincreasing sequence. Thus, the entire sequence is also bounded below by $f(\bar{x})$, and so it converges. ∎

5.3.1 Mesh Size Behavior

The behavior of the mesh size was originally characterized for NLP Problems by Torczon [130], independent of the smoothness of the objective function. It was extended to MVP problems by Audet and Dennis [8], who later adapted a Torczon lemma to provide a lower bound on the distance between mesh points at each iteration [6]. The proofs here are straightforward extensions of the latter work to MVP problems. The first lemma provides the lower bound on the distance between any two mesh points whose continuous variable values do not coincide, while the second lemma shows that the mesh size parameter is bounded above. The theorem that follows shows the key Torczon [130] result that $\liminf_{k \to +\infty} \Delta_k = 0$.

Lemma 5.7 For any integer $k \geq 0$, let u and v be any distinct points in the mesh M_k such that $u^d = v^d$. Then for any norm for which all nonzero integer vectors have norm at least 1,

$$\|u^c - v^c\| \geq \frac{\Delta_k}{\|G_i^{-1}\|}. \tag{5.8}$$

where the index i corresponds to the discrete variable values of u and v.

Proof. Using (5.4), we let $u^c = x_k^c + \Delta_k D^i z_u$ and $v^c = x_k^c + \Delta_k D^i z_v$ define the continuous part of two distinct points on M_k with both $z_u, z_v \in \mathcal{Z}_+^{|D^i|}$. Furthermore, since we assume that u and v are distinct with $u^d = v^d$, we must have $u^c \neq v^c$, and thus $z_u \neq z_v$. Then

$$\|u^c - v^c\| = \Delta_k \|D^i(z_u - z_v)\| = \Delta_k \|G_i Z_i(z_u - z_v)\| \geq \Delta_k \frac{\|Z_i(z_u - z_v)\|}{\|G_i^{-1}\|} \geq \frac{\Delta_k}{\|G_i^{-1}\|},$$

the last inequality because $Z_i(z_u - z_v)$ is a nonzero integer vector with norm greater than or equal to one. ∎

Lemma 5.8 There exists a positive integer r^u such that $\Delta_k \leq \Delta_0 \tau^{r^u}$ for any integer $k \geq 0$.

Proof. Under Assumption A1, the discrete variables can only take on a finite number of values in $L_X(x_0)$. Let $i_{\max}$ denote this number, and let $I = \{1, 2, \ldots, i_{\max}\}$. Also under Assumption A1, for each $i \in I$, let Y_i be a compact set in $\Re^{n^c}$ containing all GPS iterates whose discrete variable values correspond to $i \in I$. Let $\gamma = \max_{i \in I} \text{diam}(Y_i)$ and $\beta = \min_{i \in I} \|G_i^{-1}\|$, where diam indicates the maximum distance between any two points. If $\Delta_k > \gamma\beta$, then Lemma 5.7 (with $v = x_k$) ensures that any trial point $u \in M_k$ either satisfies $u^c = x_k^c$ or would have lied outside of $\bigcup_{i \in I} Y_i$. Then if $\Delta_k > \gamma\beta$, no more than $i_{\max}$ successful iterations will occur before Δ_k falls below $\gamma\beta$. Thus, Δ_k is bounded above by $\gamma\beta(\tau^{m_{\max}})^{i_{\max}}$, and the result follows by setting r^u large enough so that $\Delta_0 \tau^{r^u} \geq \gamma\beta(\tau^{m_{\max}})^{i_{\max}}$. ∎

Theorem 5.9 The mesh size parameters satisfy $\liminf_{k \to +\infty} \Delta_k = 0$.

Proof. (Torczon [130]) Suppose by way of contradiction that there exists a negative integer r^ℓ such that $0 < \Delta_0 \tau^{r^\ell} \leq \Delta_k$ for all integer $k \geq 0$. Combining (4.7) with Lemma 5.8 implies that for any integer $k \geq 0$, r_k takes its value from among the integers of the finite set $\{r^\ell, r^\ell + 1, \ldots, r^u\}$. Therefore, r_k and Δ_k can only take a finite number of values for all $k \geq 0$.

Since $x_{k+1} \in M_k$, (5.4) ensures that $x_{k+1}^c = x_k^c + \Delta_k D^i z_k$ for some $z_k \in \mathcal{Z}_+^{|D^i|}$ and $1 \leq i \leq i_{\max}$. By repeatedly applying this equation and substituting $\Delta_k = \Delta_0 \tau^{r_k}$, it

follows that, for any integer $N \geq 1$,

$$
\begin{aligned}
x_N^c &= x_0^c + \sum_{k=1}^{N-1} \Delta_k D^i z_k \\
&= x_0^c + \Delta_0 D^i \sum_{k=1}^{N-1} \tau^{r_k} z_k \;=\; x_0^c + \frac{p^{r^\ell}}{q^{r^u}} \Delta_0 D^i \sum_{k=1}^{N-1} p^{r_k - r^\ell} q^{r^u - r_k} z_k,
\end{aligned}
$$

where p and q are relatively prime integers satisfying $\tau = \frac{p}{q}$. Since $p^{r_k - r^\ell} q^{r^u - r_k} z_k$ is an integer for any k, it follows that the continuous part of all iterates having the same discrete variable values lies on the translated integer lattice generated by x_0^c and the columns of $\frac{p^{r^\ell}}{q^{r^u}} \Delta_0 D^i$. Moreover, the discrete part of all iterates also lies on the integer lattice $X^d \subset \mathcal{Z}^{n_d}$.

Therefore, since all iterates belong to a compact set, there must be only a finite number of different iterates, and thus one of them must be visited infinitely many times. Therefore, the mesh coarsening rule in (4.5) is only applied finitely many times, and the mesh refining rule in (4.6) is applied infinitely many times. This contradicts the hypothesis that $\Delta_0 \tau^{r^l}$ is a lower bound for the mesh size parameter. ∎

These results show the necessity of forcing the set of directions to satisfy $D^i = G_i Z_i$. Under Assumption A1, this ensures that the mesh has only a finite number of points in X, which means that there can only be a finite number of consecutive improved mesh points. Assumption A2 is included to simply ensure that this construction is maintained in the presence of linear constraints. Audet and Dennis [6] provide an example in which a different construction yields a mesh that is dense in X. In this case, Lemma 5.7 cannot be satisfied, and convergence of Δ_k to zero is not guaranteed. A sufficient condition for Assumption A2 to hold is that $G_i = I$ for each $i = 1, 2, \ldots, i_{\max}$ and that the coefficient matrix A is rational [95].

We should note also that the rationality of τ is essential for convergence. In a revised version of [4], Audet gives an example in which an irrational value for τ generates a sequence satisfying $\liminf\limits_{k \to +\infty} \Delta_k > 0$.

5.3.2 Refining Subsequences

Since Δ_k shrinks only at iterations in which no improved mesh point is found, Theorem 5.9 guarantees that the MGPS algorithm has infinitely many such iterations. Recall from Definition 4.4 that subsequences of the corresponding iterates are called *refining subsequences*. Since the concept of a limit direction must be redefined for mixed variable spaces, both definitions are now provided.

> **Definition 5.10** A subsequence of GPS mesh local optimizers $\{x_k\}_{k \in K}$ (for some subset of indices K) is said to be a *refining subsequence* if $\{\Delta_k\}_{k \in K}$ converges to zero.

> **Definition 5.11** Let $\{v_k\}_{k \in K}$ be either a refining subsequence or a corresponding subsequence of extended poll endpoints, and let $\hat{v}$ be a limit point of the subsequence. A direction $d \in D$ is said to be a *limit direction* of $\hat{v}$ if $w_k = v_k + \Delta_k(d, 0) \in X$ and $f(v_k) \leq f(w_k)$ for infinitely many $k \in K$.

The next theorem, due to Audet and Dennis [8] (but slightly reworded), identifies and proves existence of limits of refining subsequences, their discrete neighbors, and of corresponding extended poll endpoints. The result is independent of any assumptions on f.

> **Theorem 5.12** Let $L_X(x_0) = \{x \in X : f(x) \leq f(x_0)\}$. There exists a point $\hat{x} \in L_X(x_0)$ and a refining subsequence $\{x_k\}_{k \in K}$ (with associated

index set K) such that $\lim_{k \in K} x_k = \hat{x}$. Moreover, if $\mathcal{N}$ is continuous at $\hat{x}$, then there exists $\hat{y} \in \mathcal{N}(\hat{x})$ and $\hat{z} = (\hat{z}^c, \hat{y}^d) \in X$ such that

$$\lim_{k \in K} y_k = \hat{y} \quad \text{and} \quad \lim_{k \in K} z_k = \hat{z},$$

where each $z_k \in X$ is the endpoint of the EXTENDED POLL step initiated at $y_k \in \mathcal{N}(x_k)$.

Proof. (Audet and Dennis [8]). Theorem 5.9 guarantees that $\liminf_{k \to +\infty} \Delta_k = 0$; thus, there is an infinite subset of indices of mesh local optimizers $K' \subset \{k : \Delta_{k+1} < \Delta_k\}$, such that the subsequence $\{\Delta_k\}_{k \in K'}$ converges to zero. Since all iterates x_k lie in the compact set $L_X(x_0)$, there exists an infinite subset of indices $K'' \subset K'$ such that the subsequence $\{x_k\}_{k \in K''}$ converges. Let $\hat{x} \in L_X(x_0)$ be the limit point of such a subsequence.

The continuity of $\mathcal{N}$ at $\hat{x}$ guarantees that $\hat{y} \in \mathcal{N}(\hat{x}) \subset X$ is a limit point of a subsequence $y_k \in \mathcal{N}(x_k)$. Let $\hat{z} \in X$ be a limit point of the sequence $z_k \in X$ of endpoints of the EXTENDED POLL step initiated at y_k. (By definition, $z_k = y_k$ whenever the EXTENDED POLL step is not required.) Choose $K \subset K''$ to be such that both $\{y_k\}_{k \in K}$ converges to $\hat{y}$ and $\{z_k\}_{k \in K}$ is convergent, letting $\hat{z}$ denote the limit point. ∎

5.3.3 Results for the Objective Function

Theorem 5.12 defines and establishes existence of the limit points $\hat{x}$, $\hat{y}$, and $\hat{z}$. We now establish results for the objective function at these limit points. The first theorem, which Audet and Dennis originally proved by contradiction [8], establishes local optimality conditions of $\hat{x}$ with respect to its discrete neighbors. The direct proof presented here is shorter and relaxes assumptions on f as much as possible.

Theorem 5.13 Let $\hat{x}$ and $\hat{y} \in \mathcal{N}(\hat{x})$ be defined as in the statement of Theorem 5.12, such that $\mathcal{N}$ is continuous at $\hat{x}$. If f is lower semi-continuous at $\hat{x}$ with respect to the continuous variables and continuous at $\hat{y}$ with respect to the continuous variables, then $f(\hat{x}) \leq f(\hat{y})$.

Proof. From Theorem 5.12, we know that $\{x_k\}_{k \in K}$ converges to $\hat{x}$ and $\{y_k\}_{k \in K}$ converges to $\hat{y}$. Since $k \in K$ ensures that $\{x_k\}_{k \in K}$ are mesh local optimizers, we have $f(x_k) \leq f(y_k)$ for all $k \in K$, and by the assumptions of continuity and lower semi-continuity, we have $f(\hat{x}) \leq \lim_{k \in K} f(x_k) \leq \lim_{k \in K} f(y_k) = f(\hat{y})$. ∎

The remaining results in this section show convergence to points satisfying appropriate optimality conditions with respect to the continuous domain under weaker assumptions than those found in [8]. The next two theorems establish a directional optimality condition if the objective function is Lipschitz near the limit point of a refining subsequence or certain corresponding extended poll endpoints. Recall that D is defined as the finite union of mesh directions $D^i, i = 1, 2, \ldots, i_{\max}$.

Theorem 5.14 Let $\hat{x}$ be a limit point of a refining subsequence. Under Assumptions A1–A3, if f is Lipschitz near $\hat{x}$ with respect to the continuous variables, then $f^\circ(\hat{x}; (d, 0)) \geq 0$ for all limit directions $d \in D$ of $\hat{x}$.

Proof. From Definition 3.9, we have

$$f^\circ(\hat{x}; (d, 0)) = \limsup_{y \to \hat{x}, \ t \downarrow 0} \frac{f(y + t(d, 0)) - f(y)}{t} \geq \limsup_{k \in K} \frac{f(x_k + \Delta_k(d, 0)) - f(x_k)}{\Delta_k} .$$

The function f is Lipschitz, hence finite, near $\hat{x}$. Since d is a limit direction of $\hat{x}$, $x_k + \Delta_k(d, 0) \in X$ infinitely often in K, and infinitely many of the right-hand quotients are defined. All of these quotients must be nonnegative, since, for each $k \in K$, x_k is a mesh local optimizer. ∎

Theorem 5.15 Let $\hat{x}$, $\hat{y} \in \mathcal{N}(\hat{x})$, and $\hat{z}$ be defined as in the statement of Theorem 5.12, with $\mathcal{N}$ continuous at $\hat{x}$, and let $\xi > 0$ denote a lower bound on the extended poll triggers ξ_k for all k. If $f(\hat{y}) < f(\hat{x}) + \xi$ and f is Lipschitz near $\hat{z}$ with respect to the continuous variables, then $f^{\circ}(\hat{z}; (d, 0)) \geq 0$ for all limit directions $d \in D$ of $\hat{z}$.

Proof. From Definition 3.9, we have

$$f^{\circ}(\hat{z}; (d, 0)) = \limsup_{y \to \hat{z},\, t \downarrow 0} \frac{f(y + t(d, 0)) - f(y)}{t} \geq \limsup_{k \in K} \frac{f(y_k^{J_k} + \Delta_k(d, 0)) - f(y_k^{J_k})}{\Delta_k} .$$

The function f is Lipschitz, hence finite, near $\hat{z}$. Since $f(\hat{y}) < f(\hat{x}) + \xi$ ensures that extended polling was triggered around $y_k \in \mathcal{N}(x_k)$ for all $k \in K$, and since d is a limit direction of $\hat{z}$, it follows that $y_k^{J_k} + \Delta_k(d, 0) \in X$ infinitely often in K, and infinitely many of the right-hand quotients are defined. All of these quotients must be nonnegative, since, for each $k \in K$, $y_k^{J_k}$ is an extended poll endpoint. ∎

If there are no constraints on the continuous variables, then there will always be a positive spanning set D of directions that satisfy the hypothesis of Theorem 5.14. However, for bound or linear constraints, in order to guarantee the existence of any directions that satisfy the hypothesis, D must contain a sufficiently rich set of directions so that poll sets can be constructed with directions $D_k \subseteq D$ that conform to the geometry of the constraints (see Definition 4.3).

In the following two theorems, we show that $\hat{x}$ and certain $\hat{z}$ satisfy first-order necessary conditions for optimality with respect to the continuous variables, if f satisfies three conditions there (see Theorem 3.21), and if the rule for selecting poll directions conforms to X^c (Assumption A3). Recall that Definition 3.17 defines the term, D-*regular*.

Theorem 5.16 Let $\hat{x}$ be a limit point of a refining subsequence with limit directions $D(\hat{x})$, and let f satisfies the following assumptions:

1. f is Lipschitz near $\hat{x}$ with respect to the continuous variables,

2. f is differentiable at $\hat{x}$ with respect to the continuous variables,

3. f is $D(\hat{x})$-regular at $\hat{x}$ with respect to the continuous variables.

Then under Assumptions A1–A3, $\nabla^c f(\hat{x})^T w \geq 0$ for all $w \in T_{X^c}(\hat{x})$ and $-\nabla^c f(\hat{x}) \in N_{X^c}(\hat{x})$. Thus, $\hat{x}$ is a KKT point with respect to the continuous variables. Furthermore, if $X^c = \Re^{n^c}$ or if $\hat{x}^c$ lies in the interior of X^c, then $0 = \nabla^c f(\hat{x}) \in \partial^c f(\hat{x})$.

Proof. Since Assumption A3 ensures that the rule for selecting D_k conforms to X^c for some $\epsilon > 0$, and since there are finitely many linear constraints, $D(x_k) \to D(\hat{x})$, and $D(\hat{x})$ positively spans $T_{X^c}(\hat{x})$. By Theorem 5.14, we have $f^\circ(\hat{x}; (d, 0)) \geq 0$ for all $d \in D(\hat{x})$, and the result follows directly from Theorem 3.21. ∎

Theorem 5.17 Let $\hat{x}$, $\hat{y} \in \mathcal{N}(\hat{x})$, and $\hat{z}$ be defined as in the statement of Theorem 5.12, with $\mathcal{N}$ continuous at $\hat{x}$, and let $\xi > 0$ denote a lower bound on the extended poll triggers ξ_k for all k. Let $D(\hat{z})$ be the limit directions of $\hat{z}$, and suppose f satisfies the following assumptions:

1. f is Lipschitz near $\hat{z}$ with respect to the continuous variables,

2. f is differentiable at $\hat{z}$ with respect to the continuous variables,

3. f is $D(\hat{z})$-regular at $\hat{z}$ with respect to the continuous variables.

If $f(\hat{y}) < f(\hat{x}) + \xi$, then under Assumptions A1–A4, $\nabla^c f(\hat{z})^T w \geq 0$ for all $w \in T_{X^c}(\hat{z})$. Furthermore, if $X^c = \Re^{n^c}$ or $\hat{z}^c$ lies in the interior of X^c, then $0 = \nabla^c f(\hat{z}) \in \partial^c f(\hat{z})$.

Proof. Since Assumption A3 ensures that the rule for selecting D_k conforms to X^c for some $\epsilon > 0$, and since there are finitely many linear constraints, $D(z_k) \to D(\hat{z})$,

and $D(\hat{z})$ positively spans $T_{X^c}(\hat{z})$. By Theorem 5.15, we have $f^\circ(\hat{z}; (d, 0)) \geq 0$ for all $d \in D(\hat{z})$, and the result follows directly from Theorem 3.21. ∎

One special case must still be addressed; namely, the case where $f(\hat{x}) = f(\hat{y}) = f(\hat{z})$, but $\hat{y} \neq \hat{z}$. The following theorem, due to Audet and Dennis [8] (except for the very last assertion on the length of the EXTENDED POLL step, which is new) addresses this case. It shows that there are an infinite number of limit points of extended poll points with the same value as $f(\hat{x})$.

> **Theorem 5.18** Let $\hat{x}$, $\hat{y} \in \mathcal{N}(\hat{x})$, and $\hat{z}$ be defined as in the state-
> ment of Theorem 5.12, with $\mathcal{N}$ continuous at $\hat{x}$. If $f(\hat{y}) = f(\hat{x})$ and f
> is continuous at $\hat{x}$ and $\hat{y}$ with respect to the continuous variables, then
> under Assumptions A1–A4, $f(\bar{y}) = f(\hat{x})$ for any limit point $\bar{y} = (\bar{y}^c, \hat{y}^d)$ of
> the sequence of extended poll points $\{y_k^{j_k}\}_{k \in K}$. Moreover, if $\hat{y} \neq \hat{z}$, then
> there are infinitely many of these limit points $\bar{y}$, and the index J_k of the
> extended poll endpoint $y_k^{J_k}$ at iteration k satisfies $J_k \to +\infty$ (in K).

Proof. Let $\hat{y}$ in $\mathcal{N}(\hat{x})$ be such that $f(\hat{y}) = f(\hat{x})$. Let $\bar{y}$ be a limit point distinct from $\hat{y}$ and $\hat{z}$ of the sequence of extended poll points $\{y_k^{j_k}\}_{k \in K}$.

Since $f(\hat{x}) \leq f(y_k^{j_k+1}) < f(y_k^{j_k})$ for $j = 0, 1, \ldots, J_k$, and since continuity of f at $\hat{x}$ and $\hat{y}$ ensures that $\{f(y_k^0)\}_{k \in K}$ converges to $f(\hat{x})$, it follows that $f(\hat{x}) = f(\bar{y})$.

To show the second part of the result, we first let $s = \|\hat{y} - \hat{z}\|$ be the nonzero distance between $\hat{y}$ and $\hat{z}$ (which is well-defined, since both points share the same discrete components). Second, for any scalar p in the open interval $(0, s)$, we define the set

$$Y_p = \left\{ y_k^{j_k} : k \in K, \ j \in \{0, 1, \ldots, J_k\}, \ \|y_k^{j_k} - \hat{y}\| \leq p, \ \|y_k^{j_k+1} - \hat{y}\| > p\| \right\}.$$

Since $y_k^0 = y_k \to \hat{y}$ and $y_k^{J_k} = z_k \to \hat{z}$, it follows that the set Y_p contains infinitely many points for any p in $(0, s)$. Any limit point $\bar{y}_p$ of Y_p satisfies $\|\bar{y}_p - \hat{y}\| = p$, since

Δ_k converges to 0 (in K) and $y_k^{j_k+1} = y_k^{j_k} + \Delta_k d_k^{j_k}$ for some vector $d_k^{j_k} \in D_k(y_k^{j_k})$. Therefore, if $p \neq q$, then $\bar{y}_p \neq \bar{y}_q$ and the result follows.

Finally, since $\Delta_k \to 0$ (in K) and

$$
\begin{aligned}
0 \;\; &< \;\; \|\hat{z} - \hat{y}\| = \lim_{k \in K} \|y_k^{J_k} - y_k^0\| = \lim_{k \in K} \left\| \sum_{i=0}^{J_k-1} (y_k^{i+1} - y_k^i) \right\| \\
&\leq \;\; \lim_{k \in K} \sum_{i=0}^{J_k-1} \|y_k^{i+1} - y_k^i\| = \lim_{k \in K} \sum_{i=0}^{J_k-1} \|y_k^i + \Delta_k d_k^i - y_k^i\| = \lim_{k \in K} \Delta_k \sum_{i=0}^{J_k-1} \|d_k^i\|,
\end{aligned}
$$

it follows that the sum is unbounded; thus, $J_k \to +\infty$ (in K). ∎

The necessary optimality conditions shown in Theorems 5.16 and 5.17 for $\hat{x}$ and $\hat{z}$, respectively, do not necessarily hold for $\bar{y}$. In fact, an interesting counterexample is found in [8]. However, this result can be obtained for $\bar{y}$ by modifying the algorithm.

5.3.4   Stronger Results

In [130], Torczon obtains stronger convergence results for NLP problems if the POLL step is exhaustive; *i.e.*, if every feasible point in the poll set is evaluated, even if an improved mesh point has already been found. If found, the best improved mesh point becomes the next iterate. In a similar manner, Audet and Dennis [8] show that Theorem 5.17 can be strengthened if the algorithm is modified to include a stronger, but more expensive EXTENDED POLL step:

STRONG EXTENDED POLL step (at iteration k):

- Given y_k^0, set $y_k^{j+1} \in \arg \min_{y \in P_k(y_k^j)} f(y)$, $j = 0, 1, \ldots, J_k$,

- The same positive basis $D_k(y_k^j) \subset D(y_k^j)$ must be used throughout the step.

That is, instead of performing the EXTENDED POLL step only until an improvement is made, all the points in the extended poll set are evaluated, and the one with lowest

objective function value is chosen (ties are broken arbitrarily) as the poll center for the next EXTENDED POLL step (provided it is an improvement).

The final two theorems show that, if the STRONG EXTENDED POLL step is used, then the same results from Theorems 5.16 and 5.17 that apply to $\hat{x}$ and $\hat{z}$, respectively, apply also to $\bar{y}$. The first result assumes only Lipschitz continuity near $\bar{y}$, while the second assumes strict differentiability at $\bar{y}$.

> **Theorem 5.19** Let $\hat{x}$ and $\hat{y}$ be defined as in the statement of Theorem 5.12, with $\mathcal{N}$ continuous at $\hat{x}$. Suppose that $f(\hat{y}) = f(\hat{x})$, and let $\bar{y}$ be a limit point of strong extended poll points $\{y_k^{j_k}\}_{k \in K}$, where j_k is arbitrary. Under Assumptions A1–A3, if f is Lipschitz near $\bar{y}$ with respect to the continuous variables, then $f^\circ(\bar{y}; (d, 0)) \geq 0$ for all $d \in D$ for which f at a STRONG EXTENDED POLL step was evaluated infinitely often (in K).

Proof. Suppose by way of contradiction that there exists a $d \in D$ such that $f^\circ(\bar{y}; (d, 0)) < 0$. Then from Definition 3.9, we have

$$\limsup_{y \to \bar{y}, t \downarrow 0} \frac{1}{t} \left[f(y + t(d, 0)) - f(y) \right] < 0.$$

By local Lipschitz continuity, there exists an $\epsilon > 0$ such that $f(\bar{y} + t(d, 0)) < f(\bar{y})$ for all $t \in (0, \epsilon)$. Furthermore, without loss of generality, for each $t \in (0, \epsilon)$, there exists an $\epsilon_t > 0$ such that $f(\bar{y} + t(d, 0) + s(b, 0)) < f(\bar{y})$ for all $s \in (0, \epsilon_t)$ and for all $b \in \Re^n$ with $\|b\| = 1$.

Choose $k \in K$ sufficiently large such that $\|y_k^{j_k} - \bar{y}\| < \epsilon_t$ and $\Delta_k < \epsilon$. Now observe that $y_k^{j_k} = \bar{y} + \hat{s}(\hat{b}, 0)$ for some $\hat{s} \in \Re$ and $\hat{b} \in \Re^n$ with $\|\hat{b}\| = 1$. Then $\hat{s} = \hat{s}\|(\hat{b}, 0)\| = \|y_k^{j_k} - \bar{y}\| < \epsilon_t$. Use of the STRONG EXTENDED POLL step (with $j_k < J_k$) guarantees that $f(y_k^{j_k+1}) < f(y_k^{j_k} + \Delta_k(d, 0)) = f(\bar{y} + \hat{s}(b, 0) + \Delta_k(d, 0)) = f(\bar{y} + \Delta_k(d, 0) + \hat{s}(b, 0)) < f(\bar{y})$, since Δ_k is a specific case of $t \in (0, \epsilon)$ and $\hat{s} \in (0, \epsilon_t)$.

Thus, $f(y_k^{j_k+1}) < f(\bar{y}) = f(\hat{x}) \leq f(x_k)$, which contradicts the assumption that x_k is a mesh local optimizer. ∎

Theorem 5.20 Let the assumptions of Theorem 5.19 hold. Under Assumptions A1–A3, if f satisfies the assumptions of Theorem 5.16 for $\bar{y}$, then $\nabla^c f(\bar{y})^T w \geq 0$ for all $w \in T_{X^c}(\bar{y})$. If $X^c = \Re^{n^c}$ or if $\bar{y}$ lies in the interior of X^c, then $0 = \nabla^c f(\bar{y}) \in \partial^c f(\bar{y})$.

Proof. Since Theorem 5.19 applies to a set of directions that positively span $T_{X^c}(\bar{y})$, the result follows directly from it and Theorem 3.21. ∎

5.4   Summary

The new convergence results presented here show a further characterization of MGPS limit points, consistent with earlier such analyses of the GPS [6] and filter GPS [7] algorithms. In the next chapter, the FMGPS algorithm for general nonlinearly constrained MVP problems is introduced, along with convergence results developed in a similar manner. However, before doing so, it is helpful to summarize what has been proved and its significance – specifically, in the case where first-order necessary optimality conditions are proved.

First, recall that in Definition 5.3, we provided a definition of local optimality with respect to a mixed variable domain. This definition is stronger than simply requiring optimality with respect to its continuous variables and also with respect to its discrete neighbors. It also requires that the optimal point have a lower (or at least as low) objective function value than any point in a neighborhood of any of its discrete neighbors.

Given the continuity of the set-valued function $\mathcal{N}$ at $\hat{x}$ and the three assumptions on f listed in the statements of Theorems 5.16, 5.17, and 5.20, we have proved the following toward achieving the goal of local optimality:

- The limit point $\hat{x}$ achieves a weaker measure of first-order optimality, in that $\hat{x}$ satisfies first-order KKT necessary conditions for optimality with respect to its continuous variables, and it has a function value at least as low as its discrete neighbors.

- For any $\hat{y} \in \mathcal{N}(\hat{x})$ that corresponds to a subsequence that did *not* trigger extended polling infinitely often, it is trivial to show that $f(\hat{x}) < f(\hat{y})$, in which case, continuity of the objective means that $f(\hat{x}) \leq f(v)$ for all v in a neighborhood of $\hat{y}$. This is clearly a stronger condition for optimality.

- For any $\hat{y} \in \mathcal{N}(\hat{x})$ that corresponds to a subsequence that triggered extended polling infinitely often, the limit point $\hat{z}$ satisfies $f(\hat{x}) \leq f(\hat{z}) \leq f(\hat{y})$, and $\hat{z}$ satisfies the KKT necessary conditions for optimality with respect to its continuous variables. Thus, f has no feasible descent directions at $\hat{z}$. This is about as close as we can come to establishing and satisfying a characterization of first-order optimality conditions for mixed variable problems.

- The STRONG EXTENDED POLL STEP takes care of the special case where $f(\hat{y}) = f(\hat{x})$, but $\hat{y} \neq \hat{z}$. In this case, under the normal algorithm, it is possible to have a limit point $\bar{y}$ of intermediate extended poll points which has descent directions not detected by the algorithm. Although this scenario is highly unlikely in practice, a simple example in which this occurs can be found in [8]. The stronger version of the algorithm alleviates this problem.

Chapter 6

GPS for General Constrained MVP Problems

In this chapter, we introduce a new class of filter GPS algorithms for mixed variable optimization problems with general nonlinear constraints. It is constructed by merging the ideas of the previous two chapters in a way that encompasses the GPS algorithms described there, so that the convergence results of Chapters 4 and 5 become corollaries of the results presented in this chapter.

As discussed in Chapters 1 and 5, the MVP problem formulation is complicated by the fact that changes in the discrete variables can mean a change in the constraints, and even a change in problem dimension. Thus, we denote n^c and n^d as the *maximum* dimensions of the continuous and discrete variables, respectively, and we partition each point $x = (x^c, x^d)$ into continuous variables $x^c \in \Re^{n^c}$ and discrete variables $x^d \in \mathcal{Z}^{n^d}$. Again, we adopt the convention of ignoring unused variables.

The problem under consideration, which is restated from (1.1), can be expressed as follows:

$$
\begin{aligned}
\min_{x \in X} \quad & f(x) \\
\text{s. t.} \quad & C(x) \leq 0,
\end{aligned}
\tag{6.1}
$$

where $f : X \rightarrow \Re \cup \{\infty\}$, and $C : X \rightarrow (\Re \cup \{\infty\})^p$ with $C = (C_1, C_2, \ldots, C_p)^T$. The domain $X = X^c \times X^d$ is partitioned into continuous and discrete variable spaces $X^c \subseteq \Re^{n^c}$ and $X^d \subseteq \mathcal{Z}^{n^d}$, respectively, where X^c is defined by a finite set of bound and linear constraints, dependent on the values of x^d. That is,

$$
X^c(x^d) \quad = \quad \{x^c \in \Re^{n^c} : \ell(x^d) \leq A(x^d)x^c \leq u(x^d)\},
\tag{6.2}
$$

where $A(x^d) \in \Re^{m^c \times n^c}$ is a real matrix, $\ell(x^d), u(x^d) \in (\Re \cup \{\pm\infty\})^{n^c}$, and $\ell(x^d) \leq u(x^d)$ for all values of x^d.

Once again, although we allow equality constraints in the theory, they cause significant numerical difficulties. Thus, in practice, variables ought to be eliminated until all the linear constraints can be expressed in a way that satisfies $\ell(x^d) < u(x^d)$ for all values of x^d.

Note that this formulation is indeed a generalization of the standard NLP problem, in that, if $n^d = 0$, then the problem reduces to a standard NLP problem, in which ℓ, A, u, and C do not change. For convenience, we now omit the explicit dependence on x^d in this chapter, even though it is understood for MVP problems.

6.1 The FMGPS Algorithm

For the new class of algorithms, since we are adding the filter construct to the mixed variable algorithm, we must redefine the mesh and poll set in terms of a poll center, as opposed to a current iterate x_k (see Chapter 5). That is, at each iteration k, the poll center $p_k \in \{p_k^F, p_k^I\}$ is chosen as either the incumbent best feasible point p_k^F or the incumbent least infeasible point p_k^I. The mesh is constructed as the direct product of X^d with the union of a finite number of lattices, but translated from the poll center; *i.e.,*

$$M_k = X^d \times \bigcup_{i=1}^{i_{\max}} \{p_k^c + \Delta_k D^i z \in X^c : z \in \mathcal{Z}_+^{|D^i|}\}, \tag{6.3}$$

where D^i denotes a set of positive spanning directions corresponding to the i-th set of discrete variable values. Each set D^i (which is represented as a matrix whose columns are members of the set) is constructed in exactly the same manner as described in Section 5.1, and is often written as $D(x)$ to indicate that it corresponds to the discrete variable values of $x \in X$.

The set of discrete neighbors is again constructed in terms of a set-valued function $\mathcal{N} : X \to 2^X$, with the convention that, for each $x \in X$, $x \in \mathcal{N}(x)$, which is finite. Continuity of $\mathcal{N}$ (see Definition 5.2) will be assumed at certain limit points of the algorithm in order to obtain some of the convergence results that follow.

The poll set centered at p_k is defined as

$$P_k(p_k) = \{p_k + \Delta_k(d, 0) \in X : d \in D_k(p_k)\}, \tag{6.4}$$

where the set $D_k(p_k) \subset D^i$ for some $1 \leq i \leq i_{\max}$ is a set of positive spanning set of poll directions.

Because the filter seeks a better point with respect to either of the two functions (the objective function f and the constraint violation function h), a change must be made to the rule for selecting discrete neighbors, about which to perform an EXTENDED POLL step. Recall that in the MGPS algorithm, extended polling is performed around any discrete neighbor whose objective function value is sufficiently close to that of the current iterate (*i.e.*, "almost" an improved mesh point). With the addition of nonlinear constraints to the problem, we now require a notion of a discrete neighbor "almost" generating a new incumbent best feasible point or least infeasible point.

While this issue has by no means a single workable approach, the implementation here has the desirable property of being a generalization of the one described in Chapter 5. At iteration k, let $f_k^F = f(p_k^F)$ denote the objective function value of the incumbent best feasible point, and let $h_k^I = h(p_k^I)$ be the constraint violation function value of the incumbent least infeasible point. Given current poll center p_k and user-specified extended poll triggers $\xi_k^f \leq \xi > 0$ and $\xi_k^h \leq \xi > 0$ for f and h, respectively (where ξ is a positive constant), we perform an EXTENDED POLL step around any discrete neighbor $y_k \in \mathcal{N}(p_k)$ satisfying either $0 < h_k^I < h(y_k) < \min(h_k^I + \xi_k^h, h_{\max})$,

or $h(y_k) = 0$ with $f_k^F < f(y_k) < f_k^F + \xi_k^f$. The extended poll triggers ξ_k^f and ξ_k^h can also be set according to the categorical variable values associated with the current poll center, but this dependency is not included in the notation, so as not to obfuscate the ideas presented here.

Similar to the GMVP algorithm of Chapter 5, the EXTENDED POLL STEP generates a sequence of EXTENDED POLL centers $\{y_k^j\}_{j=0}^{J_k}$, beginning with $y_k^0 = y_k$ and ending with *extended poll endpoint*, $y_k^{J_k} = z_k$.

Thus, at iteration k, the set of all points evaluated in the EXTENDED POLL step, denoted $\mathcal{X}_k(\xi_k^f, \xi_k^h)$, is

$$\mathcal{X}_k(\xi_k^f, \xi_k^h) = \bigcup_{y \in \mathcal{N}_k^f \cup \mathcal{N}_k^h} \mathcal{E}(y) \tag{6.5}$$

where $\mathcal{E}(y)$ denotes the set of extended poll points, and

$$\mathcal{N}_k^f = \{y \in \mathcal{N}(p_k) : h(y) = 0, f_k^F \leq f(y) \leq f_k^F + \xi_k^f\}, \tag{6.6}$$

$$\mathcal{N}_k^h = \{y \in \mathcal{N}(p_k) : 0 < h_k^I < h(y) < \min(h_k^I + \xi_k^h, h_{\max})\}. \tag{6.7}$$

The set of trial points is defined as $T_k = S_k \cup P_k(p_k) \cup \mathcal{N}(p_k) \cup \mathcal{X}_k(\xi_k^f, \xi_k^h)$, where S_k is the finite set of mesh points evaluated during the SEARCH step. Similar to the Filter GPS algorithm for NLP problems, described in Section 4.2, the following definitions define the two outcomes of the SEARCH, POLL, and EXTENDED POLL steps.

Definition 6.1 Let T_k denote the set of trial points to be evaluated at iteration k, and let $\overline{\mathcal{F}}_k$ denote the set of filtered points described by (4.8). A point $y \in T_k$ is said to be an *unfiltered point* if $y \notin \overline{\mathcal{F}}_k$.

Definition 6.2 Let $P_k(p_k)$ denote the poll set centered at the point p_k, and let $\overline{\mathcal{F}}_k$ denote the set of filtered points described by (4.8). The point p_k is said to be a *mesh isolated filter point* if $P_k(p_k) \cup \mathcal{N}(p_k) \cup \mathcal{X}(\xi_k^f, \xi_k^h) \subset \overline{\mathcal{F}}_k$. If $p_k \in \{p_k^F, p_k^I\}$, then p_k is said to be a *mesh isolated poll center*.

Figure 6.1 depicts a filter on a bi-loss graph, in which the best feasible and least infeasible solutions are indicated, and the feasible solutions lie on the vertical axis (labelled f). The dashed lines have been added to indicate the areas for which an EXTENDED POLL step is triggered. If a feasible discrete neighbor has an objective function value higher on the axis than the current incumbent, but lower than the horizontal dashed line, an EXTENDED POLL step is performed around this discrete neighbor. Similarly, an EXTENDED POLL step is performed if a filtered discrete neighbor lies to the right of the current least infeasible solution, but left of the vertical dashed line.
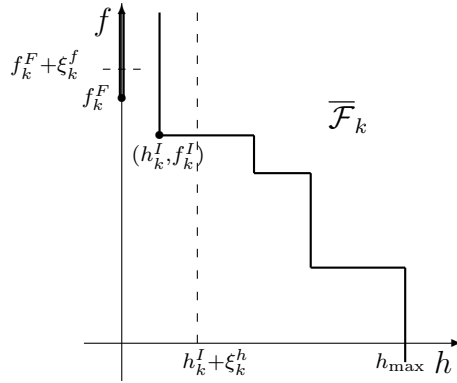
Figure 6.1   MVP Filter and Extended Poll Triggers.

Recall that in the Filter GPS algorithm presented in Chapter 4, the goal of each iteration is to find an unfiltered point. This is the case here as well, but the details of when to continue an EXTENDED POLL step must be generalized from the simple decrease condition in f under which the MGPS algorithm operates. More specifically, if the EXTENDED POLL step finds an unfiltered point, it is added to the filter, the poll center is updated (if appropriate), and the mesh is coarsened according to the rule in (4.5). If the EXTENDED POLL step fails to find a new point y satisfying $y \in \mathcal{N}_k^f \cup \mathcal{N}_k^h$,

then the current incumbent poll center p_k is declared to be a mesh isolated filter point, the current poll center is retained, and the mesh is refined according to the rule in (4.6). However, the unanswered question is how to treat extended poll points that are filtered, yet still belong to $\mathcal{N}_k^f$ or $\mathcal{N}_k^h$.

In order to handle this case, we establish the notion of a temporary *local filter*. At iteration k, for each discrete neighbor y_k, a local filter $\mathcal{F}_k^L(y_k)$ is simply a filter relative to the current EXTENDED POLL step. It is populated initially with only the point y_k and with $h_{\max}^L = \min(h_k^I + \xi_k^h, h_{\max})$. As with the MGPS algorithm, the extended poll sequence $\{y_k^j\}_{j=1}^{J_k}$ begins with $y_k^0 = y_k$ and ends with $z_k = y_k^{J_k}$, where each y_k^j is the poll center of the local filter – chosen either as the best feasible or least infeasible point, relative to the local filter. Extended polling with respect to y_k proceeds, with points being added to the local filter as appropriate, until no more unfiltered mesh points can be found with respect to the new local filter, or until an unfiltered point is found with respect to the main filter. Once either of these conditions is satisfied, the EXTENDED POLL step ends, and the main filter is appropriately updated with the points of the local filter, which is then discarded. The mesh size parameter Δ_k, which has been kept constant throughout the step, is then updated, depending on the success of the SEARCH, POLL, and EXTENDED POLL steps in finding an unfiltered point with respect to the main filter.

The EXTENDED POLL step is summarized by the algorithm in Figure 6.2, and the FMGPS Algorithm is summarized in Figure 6.3

6.2   Convergence Results

The convergence properties of the new algorithm are now presented in four sections. First, the behavior of the mesh size parameter Δ_k will be shown to have the same behavior as previous results, and a general characterization of limit points of certain

---

EXTENDED POLL Step at Iteration k

Input: Current poll center p_k, filter $\mathcal{F}_k$, and extended poll triggers ξ_k^f and ξ_k^h.

For each discrete neighbor $y_k \in \mathcal{N}_k^f \cup \mathcal{N}_k^h$ (see (6.6) and (6.7)), do the following:

- Initialize local filter $\mathcal{F}_k^L$ with y_k and $h_{\max}^L = \min\{h_k^I + \xi_k^h, h_{\max}\}$. Set $y_k^0 = y_k$.

- For $j = 0, 1, 2, \ldots$

  1. Evaluate f and h at points in $P_k(y_k^j)$ until a point w is found that is unfiltered with respect to $\mathcal{F}_k^L$, or until done.
  2. If no point $w \in P_k(y_k^j)$ is unfiltered with respect to $\mathcal{F}_k^L$, then go to **Next**.
  3. If a point w is unfiltered with respect to $\mathcal{F}_k$, set $x_{k+1} = w$ and Quit.
  4. If w is filtered with respect to $\mathcal{F}_k$, but unfiltered with respect to $\mathcal{F}_k^L$, then update $\mathcal{F}_k^L$ to include w, and compute new extended poll center y_k^{j+1}.

- **Next:** Discard $\mathcal{F}_k^L$ and process next $y_k \in \mathcal{N}_k^f \cup \mathcal{N}_k^h$.

Figure 6.2 Extended Poll Step for the FMGPS Algorithm

subsequences is given. Results for the constraint violation function and for the objective function follow, similar to those found in [7]. Finally, stronger results for a more specific implementation of the new algorithm are provided. These mimic those found in Section 5.3.4, but apply to the more general MVP problem with nonlinear constraints.

Filter Mixed Variable Generalized Pattern Search – FMGPS

Initialization: Let x_0 be an undominated point of a set of initial solutions. Include all these points in the filter $\mathcal{F}_0$, with $h_{\max} > h(x_0)$. Fix $\xi > 0$ and $\Delta_0 > 0$.

For $k = 0, 1, 2, \ldots$, perform the following:

1. Update poll center $p_k \in \{p_k^F, p_k^I\}$, extended poll triggers $\xi_k^f \geq \xi$ and $\xi_k^h \geq \xi$.

2. Compute incumbent values $f_k^F = f(p_k^F), h_k^I = h(p_k^I), f_k^I = f(p_k^I)$.

3. SEARCH **step:** Employ some finite strategy seeking an unfiltered mesh point $x_{k+1} \notin \overline{\mathcal{F}}_k$.

4. POLL **step:** If the SEARCH step did not find an unfiltered point, evaluate f and h at points in the poll set $P_k(p_k) \cup \mathcal{N}(p_k)$ until an unfiltered mesh point $x_{k+1} \notin \overline{\mathcal{F}}_k$ is found, or until done.

5. EXTENDED POLL **step:** If SEARCH and POLL did not find an unfiltered point, execute the algorithm in Figure 6.2 to continue looking for $x_{k+1} \notin \overline{\mathcal{F}}_k$.

6. **Update:** If SEARCH, POLL, or EXTENDED POLL finds an unfiltered point, Update filter $\mathcal{F}_{k+1}$ with x_{k+1}, and set $\Delta_{k+1} \geq \Delta_k$ according to (4.5); Otherwise, set $\mathcal{F}_{k+1} = \mathcal{F}_k$, and set $\Delta_{k+1} < \Delta_k$ according to (4.6).

Figure 6.3   FMGPS Algorithm

We make the following assumptions, consistent with those of previous GPS algorithms:

A1: All iterates $\{x_k\}$ produced by the FMGPS algorithm lie in a compact set.

A2: For each fixed set of discrete variable values x^d, the corresponding set of directions $D^i = G_i Z_i$, as defined in (5.3), includes tangent cone generators for every point in $X^c(x^d)$.

A3: The rule for selecting directions D_k conforms to X^c for some $\epsilon > 0$ (see Definition 4.3).

A4: The discrete neighbors always lie on the mesh; i.e., $\mathcal{N}(x_k) \subset M_k$ for all k.

6.2.1 Mesh Size Behavior and Limit Points

Since the behavior of the mesh size is governed by Assumptions A1–A4, the following three results are identical to Lemmas 5.7 and 5.8, and Theorem 5.9 in Chapter 5. The proofs are not repeated here.

Lemma 6.3 For any integer $k \geq 0$, let u and v be any distinct points in the mesh M_k such that $u^d = v^d$. Then for any norm for which all nonzero integer vectors have norm at least 1,

$$\|u^c - v^c\| \geq \frac{\Delta_k}{\|G_i^{-1}\|}.$$

where the index i corresponds to the discrete variable values of u and v.

Lemma 6.4 There exists a positive integer r^u such that $\Delta_k \leq \Delta_0 \tau^{r^u}$ for any $k \geq 0$.

Theorem 6.5 The mesh size parameters satisfy $\liminf_{k \to +\infty} \Delta_k = 0$.

We will again be interested in refining subsequences; thus the following two definitions are provided. The first is identical to Definition 4.15, while the second is modified from that of Definition 4.16 to include additional limit points.

Definition 6.6 A subsequence of mesh isolated poll centers $\{p_k\}_{k \in K}$ (for some subset of indices K) is said to be a *refining subsequence* if $\{\Delta_k\}_{k \in K}$ converges to zero.

Definition 6.7 Let $\{v_k\}_{k \in K}$ be either a refining subsequence or a corresponding subsequence of extended poll endpoints, and let $\hat{v}$ be a limit point of the subsequence. A direction $d \in D$ is said to be a *limit direction* of $\hat{v}$ if $v_k + \Delta_k(d, 0)$ belongs to X and is filtered for infinitely many $k \in K$.

The following theorem of Audet and Dennis [8] establishes the existence of limit points of specific subsequences of interest. Its proof is omitted, since it is essentially identical to that of Theorem 5.12.

Theorem 6.8 There exists a point $\hat{p} \in \{x \in X : f(x) \leq f(x_0)\}$ and a refining subsequence $\{p_k\}_{k \in K}$ (with associated index set K) such that $\lim_{k \in K} p_k = \hat{p}$. Moreover, if $\mathcal{N}$ is continuous at $\hat{p}$, then there exists $\hat{y} \in \mathcal{N}(\hat{p})$ and $\hat{z} = (\hat{z}^c, \hat{y}^d) \in X$ such that

$$\lim_{k \in K} y_k = \hat{y} \quad \text{and} \quad \lim_{k \in K} z_k = \hat{z},$$

where each $z_k \in X$ is the endpoint of the EXTENDED POLL step initiated at $y_k \in \mathcal{N}(p_k)$.

The notation in Theorem 6.8 describing specific subsequences and their limit points will be retained and used throughout the remainder of this chapter.

6.2.2 Results for the Constraint Violation Function

Theorem 6.8 defines and establishes existence of the limit points $\hat{p}$, $\hat{y}$, and $\hat{z}$. While the next result applies to more general limit points of the algorithm, the remainder of the results in this section apply to these specific limit points. This pattern will be repeated in Section 6.2.3 as well. This first result, which is similar to a theorem in [6] for f, requires a very mild condition on h. Note that this result will not hold for f without an additional assumption because there is no guarantee that any subsequence of objective function values is nonincreasing.

Theorem 6.9 If h is lower semicontinuous with respect to the continuous variables at a limit point $\bar{p}$ of poll centers $\{p_k\}$, then $\lim_k h(p_k)$ exists and is greater than or equal to $h(\bar{p}) \geq 0$. If h is continuous at every limit point of $\{p_k\}$, then every limit point has the same function value.

Proof. If $h(\bar{p}) = 0$, the result follows trivially. Now let $h(\bar{p}) > 0$. Then $\bar{p}$ is a limit point of a sequence of least infeasible points p_k^I, which is monotonically nonincreasing. Since h is lower semicontinuous at $\bar{p}$, we know that for any subsequence $\{p_k\}_{k \in K}$ of poll centers that converges to $\bar{p}$, $\liminf_{k \in K} h(p_k) \geq h(\bar{p}) \geq 0$. But the subsequence of constraint violation function values at p_k^I is a subsequence of a nonincreasing sequence. Thus, the entire sequence is also bounded below by $h(\bar{p})$, and so it converges. ∎

We now characterize the limit points of Theorem 6.8 with respect to the constraint violation function h. The following theorem establishes the local optimality of h at $\hat{p}$ with respect to its discrete neighbors. The proof is nearly identical to that of Theorem 5.13.

> **Theorem 6.10** Let $\hat{p}$ and $\hat{y} \in \mathcal{N}(\hat{p})$ be defined as in the statement of Theorem 6.8, with $\mathcal{N}$ is continuous at $\hat{x}$. If h is lower semi-continuous at $\hat{p}$ with respect to the continuous variables and continuous at $\hat{y}$ with respect to the continuous variables, then $h(\hat{p}) \leq h(\hat{y})$.

Proof. From Theorem 5.12, we know that $\{p_k\}_{k \in K}$ converges to $\hat{p}$ and $\{y_k\}_{k \in K}$ converges to $\hat{y}$. Since $k \in K$ ensures that $\{p_k\}_{k \in K}$ are mesh isolated poll centers, we have $h(p_k) \leq h(y_k)$ for all $k \in K$, and by the assumptions of continuity and lower semi-continuity, we have $h(\hat{p}) \leq \lim_{k \in K} h(p_k) \leq \lim_{k \in K} h(y_k) = h(\hat{y})$. ∎

The next two results establish a directional optimality condition for h at $\hat{p}$ and at certain $\hat{z}$ with respect to the continuous variables.

> **Theorem 6.11** Let $\hat{p}$ be a limit point of a refining subsequence. Under Assumptions A1–A4, if h is Lipschitz near $\hat{p}$ with respect to the continuous variables, then $h^\circ(\hat{p}; (d, 0)) \geq 0$ for all limit directions $d \in D(\hat{p})$ of $\hat{p}$.

Proof. Let $\{p_k\}_{k \in K}$ be a refining subsequence with limit point $\hat{p}$ and let $d \in D(\hat{p})$ be a limit direction of $\hat{p}$). From Definition 3.9, we have

$$h^{\circ}(\hat{p};(d,0)) \;=\; \limsup_{y \to \hat{p},\ t \downarrow 0} \frac{h(y + t(d,0)) - h(y)}{t} \;\geq\; \limsup_{k \in K} \frac{h(p_k + \Delta_k(d,0)) - h(p_k)}{\Delta_k} \;.$$

The function h is Lipschitz, hence finite, near $\hat{p}$. Since points that are infeasible with respect to X are not evaluated by the algorithm, the assumption of d being a limit direction of $\hat{p}$ ensures that infinitely many right-hand quotients are defined. All of these quotients must be nonnegative, or else the corresponding POLL step would have found an unfiltered point, a contradiction. ∎

Theorem 6.12 Let $\hat{p}$, $\hat{y} \in \mathcal{N}(\hat{p})$, and $\hat{z}$ be defined as in the statement of Theorem 6.8, with $\mathcal{N}$ continuous at $\hat{p}$, and let $\xi > 0$ denote a lower bound on the extended poll triggers ξ_k^f and ξ_k^h for all k. If $h(\hat{y}) < h(\hat{p}) + \xi$ and h is Lipschitz near $\hat{z}$ with respect to the continuous variables, then $h^{\circ}(\hat{z};(d,0)) \geq 0$ for all limit directions $d \in D(\hat{z})$ of $\hat{z}$.

Proof. From Definition 3.9, we have

$$h^{\circ}(\hat{z};(d,0)) \;=\; \limsup_{y \to \hat{z},\ t \downarrow 0} \frac{h(y + t(d,0)) - f(y)}{t} \;\geq\; \limsup_{k \in K} \frac{h(z_k + \Delta_k(d,0)) - h(z_k)}{\Delta_k} \;.$$

Now, h is Lipschitz, hence finite, near $\hat{z}$. Since $h(\hat{y}) < h(\hat{p}) + \xi$ ensures that extended polling was triggered around $y_k \in \mathcal{N}(p_k)$ for all $k \in K$, and since d is a limit direction of $\hat{z}$, it follows that $z_k + \Delta_k(d,0) \in X$ infinitely often in K, and infinitely many of the right-hand quotients are defined. All of these quotients must be nonnegative, since for $k \in K$, z_k is an extended poll endpoint. ∎

For bound or linear constraints, in order to guarantee the existence of limit directions, for which Theorem 6.11 applies, each $D^i \subset D, i = 1, 2, \ldots, i_{\max}$ is constructed in accordance with the algorithm given in Figure 4.3 to generate a sufficiently rich

set of directions to ensure conformity to X^c (see Definition 4.3), consistent with Assumption A3.

The next two key theorems establish conditions on h at $\hat{p}$ and certain $\hat{z}$ to satisfy a first-order optimality condition with respect to the continuous variables. Recall that the meaning of *D-regularity* is given in Definition 3.17.

Theorem 6.13   Let $\hat{p}$ be the limit of a refining subsequence with limit directions $D(\hat{p})$, and suppose h satisfies the following conditions:

1. h is Lipschitz near $\hat{p}$ with respect to the continuous variables,

2. h is differentiable at $\hat{p}$ with respect to the continuous variables,

3. h is $D(\hat{p})$-regular at $\hat{p}$ with respect to the continuous variables.

Then under Assumptions A1–A3, $\nabla^c h(\hat{p})^T w \geq 0$ for all $w \in T_{X^c}(\hat{p})$. Moreover, if $X^c = \Re^{n^c}$, or if $\hat{p}$ lies in the interior of X^c, then $0 = \nabla^c h(\hat{p}) \in \partial^c h(\hat{p})$.

Proof. Since Assumption A3 ensures that the rule for selecting D_k conforms to X^c for some $\epsilon > 0$, and since there are finitely many linear constraints, $D(p_k) \to D(\hat{p})$, and $D(\hat{p})$ positively spans $T_{X^c}(\hat{p})$. Theorem 6.11 guarantees that $h^\circ(\hat{p}, (d, 0)) \geq 0$ for all $d \in D(\hat{p})$, and the result follows directly from Theorem 3.21. ∎

Theorem 6.14   Let $\hat{p}, \hat{y} \in \mathcal{N}(\hat{p})$, and $\hat{z}$ be defined as in the statement of Theorem 6.8, with $\mathcal{N}$ continuous at $\hat{p}$, and let $\xi > 0$ denote a lower bound on the extended poll triggers ξ_k^f and ξ_k^h for all k. Let $D(\hat{z})$ denote the limit directions of $\hat{z}$, and suppose h satisfies the following assumptions:

1. h is Lipschitz near $\hat{z}$ with respect to the continuous variables,

2. h is differentiable at $\hat{z}$ with respect to the continuous variables,

3. h is $D(\hat{z})$-regular at $\hat{z}$ with respect to the continuous variables.

If $h(\hat{y}) < h(\hat{p}) + \xi$, then under Assumptions A1–A4, $\nabla^c h(\hat{z})^T w \geq 0$ for all $w \in T_{X^c}(\hat{z})$. Furthermore, if $X^c = \Re^{n^c}$ or $\hat{z}^c$ lies in the interior of X^c, then $0 = \nabla^c h(\hat{z}) \in \partial^c h(\hat{z})$.

Proof. Since Assumption A3 ensures that the rule for selecting D_k conforms to X^c for some $\epsilon > 0$, and since there are finitely many linear constraints, $D(z_k) \to D(\hat{z})$, and $D(\hat{z})$ positively spans $T_{X^c}(\hat{z})$. Theorem 6.12 ensures that $h^\circ(\hat{z}; (d, 0)) \geq 0$ for all $d \in D(\hat{z})$, and the result follows directly from Theorem 3.21. ∎

Recall that in Theorem 5.18, we showed that, for any discrete neighbor $\hat{y} \in \mathcal{N}(\hat{p})$ satisfying $f(\hat{y}) = f(\hat{p})$, we have $f(\bar{y}) = f(\hat{p})$, where $\bar{y}$ is any limit point of the sequence of extended poll points $\{y_k^{j_k}\}_{k \in K}$. The following theorem shows a similar result for h, but requires the limit points $\hat{p}$ and $\hat{y}$ to be infeasible. The result does not hold if the limit points are feasible because there is no guarantee that $\bar{y}$ must be feasible, particularly if the extended poll centers are chosen infinitely often to be least infeasible points with respect to the local filter.

Theorem 6.15 Let $\hat{p}$, $\hat{y} \in \mathcal{N}(\hat{p})$, and $\hat{z}$ be defined as in the statement of Theorem 6.8, with $\mathcal{N}$ continuous at $\hat{p}$, and suppose $h(\hat{y}) = h(\hat{p}) > 0$. If h is continuous at $\hat{p}$ and $\hat{y}$ with respect to the continuous variables, then under Assumptions A1–A4, either $h(\bar{y}) = 0$ or $h(\bar{y}) = h(\hat{p})$ for any limit point $\bar{y} = (\bar{y}^c, \hat{y}^d)$ of the sequence of extended poll centers $\{y_k^{j_k}\}_{k \in K}$.

Moreover, if $\hat{y} \neq \hat{z}$, then there are infinitely many of these limit points $\bar{y}$, and the index J_k of the extended poll endpoint $y_k^{J_k}$ at iteration k satisfies $J_k \to +\infty$ (in K).

Proof. Let $\hat{y}$ in $\mathcal{N}(\hat{p})$ satisfy $h(\hat{y}) = h(\hat{p})$. Let $\bar{y}$ be a limit point of the sequence of extended poll points $\{y_k^{j_k}\}_{k \in K}$, distinct from $\hat{y}$ and $\hat{z}$.

If $h(\bar{p}) > 0$, then the hypothesis that $h(\hat{p}) = h(\hat{y}) > 0$ means that, for all sufficiently large $k \in K$, we have $0 < h(\hat{p}) \leq h(y_k^{j_k}) < h(y_k^{j_k-1})$ for $j = 1, 2, \ldots, J_k$. Since the subsequence $\{h(y_k^0)\}_{k \in K}$ converges to $h(\hat{p})$, we conclude that $h(\hat{p}) = h(\bar{y})$. The remainder of the proof is identical to that of Theorem 5.18. ∎

In section 6.2.4, we will show a stronger version of the algorithm which will guarantee for $\bar{y}$ the same optimality conditions for h that apply to $\hat{p}$ and $\hat{z}$.

6.2.3 Results for the Objective Function

We now address the properties of certain limit points with respect to the objective function f. Unfortunately, in order to obtain results for f that are similar to those for h, an additional hypothesis must be added to most of the theorems that follow. Additionally, convergence to a KKT point (with respect to the continuous variables) cannot be guaranteed, but we will show a similar result to Theorem 4.20, in which a cone is identified whose polar contains the normal cone.

The first result, under very mild conditions, is similar to previous results (see Theorems 5.6 and 6.9), but requires polling to be centered at the best feasible point at all but finitely many iterations.

Theorem 6.16 Under Assumption A1, there exists at least one limit point $\bar{p}$ of the iteration sequence $\{p_k\}$ of poll centers. If f is lower semi-continuous at $\bar{p}$ with respect to the continuous variables, h is continuous at $\bar{p}$ with respect to the continuous variables, and $p_k = p_k^F$ for all but finitely many k, then $\lim_k f(p_k)$ exists and is greater than or equal to $f(\bar{p})$, which is finite. If f is continuous at every limit point of $\{p_k\}$, then every limit point has the same function value.

Proof. First, $p_k = p_k^F$, hence $h(p_k) = 0$, for all but finitely many k. Thus, f is nonincreasing, for all sufficiently large k. Since f is lower semicontinuous at $\bar{p}$, we know that for any subsequence $\{p_k\}_{k \in K}$ of poll centers converging to $\bar{p}$, $\liminf_{k \in K} f(p_k) \geq f(\bar{p})$. But the subsequence of function values is a subsequence of a nonincreasing sequence (for sufficiently large k). Thus, for sufficiently large k, the sequence is also bounded below by $f(\bar{p})$, and so it converges. $\blacksquare$

The remainder of this section contains results for the limit points described by Theorem 6.8. Each theorem contains an additional necessary hypothesis that, for infinitely many iterations of the specified subsequence, trial points must be filtered by the poll center (or extended poll endpoint), rather than a different filter point.

The following result, which is similar to Theorems 5.13 and 6.10, establishes optimality conditions with respect to the discrete set of neighbors.

> **Theorem 6.17** Let $\hat{p}$ and $\hat{y} \in \mathcal{N}(\hat{p})$ be defined as in the statement of Theorem 6.8, with $\mathcal{N}$ continuous at $\hat{x}$. If f is lower semi-continuous at $\hat{p}$ and $\hat{y}$ with respect to the continuous variables, and if $f(p_k) \leq f(y_k)$ for infinitely many $k \in K$, then $f(\hat{p}) \leq f(\hat{y})$.

Proof. From Theorem 6.8, we know that $\{p_k\}_{k \in K}$ converges to $\hat{p}$ and $\{y_k\}_{k \in K}$ converges to $\hat{y}$. Without loss of generality, we may assume that $h(p_k) < h_{\max}$ for all $k \in K$. Then, since $f(p_k) \leq f(y_k)$ for infinitely many $k \in K$, we have by the assumptions of continuity and lower semi-continuity, that $f(\hat{p}) \leq \lim_{k \in K} f(p_k) \leq \lim_{k \in K} f(y_k) = f(\hat{y})$. $\blacksquare$

The next two results establish conditions under which certain Clarke generalized directional derivatives are nonnegative. The first theorem applies to $\hat{p}$, while the second applies to some $\hat{z}$. As before, these theorems require the additional hypothesis

that the incumbent poll center or extended poll endpoint, rather than a different filter point, filter the trial points infinitely often in the subsequence.

Theorem 6.18   Let $\hat{p}$ be a limit point of a refining subsequence $\{p_k\}_{k \in K}$, and let $d \in D$ be a limit direction of $\hat{p}$. Under Assumptions A1–A4, if f is Lipschitz near $\hat{p}$ with respect to the continuous variables, and $f(p_k) \leq f(p_k + \Delta_k(d,0))$ for infinitely many $k \in K$, then $f^\circ(\hat{p}; (d,0)) \geq 0$.

Proof. From Definition 3.9, we have that

$$f^\circ(\hat{p}; (d,0)) \;=\; \limsup_{y \to \hat{p}, \; t \downarrow 0} \frac{f(y + t(d,0)) - f(y)}{t} \;\geq\; \limsup_{k \in K} \frac{f(p_k + \Delta_k(d,0)) - f(p_k)}{\Delta_k},$$

which is nonnegative, since an infinite number of terms in the right-hand quotient are nonnegative.   ∎

Theorem 6.19   Let $\hat{p}$, $\hat{y} \in \mathcal{N}(\hat{p})$, $\hat{z}$, and z_k be defined as in the statement of Theorem 6.8, with $\mathcal{N}$ continuous at $\hat{p}$, and let $d \in D$ be a limit direction for $\hat{z}$. Suppose that $f(\hat{y}) < f(\hat{p}) + \xi$, where $\xi > 0$ is a lower bound on the extended poll triggers ξ_k^f and ξ_k^h for all k. Under Assumptions A1–A4, if f is Lipschitz near $\hat{z}$ with respect to the continuous variables, and $f(z_k) \leq f(z_k + \Delta_k(d,0))$ for infinitely many $k \in K$, then $f^\circ(\hat{z}; (d,0)) \geq 0$.

Proof. From Definition 3.9, we have that

$$f^\circ(\hat{z}; (d,0)) \;=\; \limsup_{y \to \hat{z}, \; t \downarrow 0} \frac{f(y + t(d,0)) - f(y)}{t} \;\geq\; \limsup_{k \in K} \frac{f(z_k + \Delta_k(d,0)) - f(z_k)}{\Delta_k},$$

which is nonnegative, since an infinite number of terms in the right-hand quotient are nonnegative.   ∎

The next two results describe the optimality conditions for f at $\hat{p}$ and at certain $\hat{z}$ under the assumptions of local Lipschitz continuity, differentiability, and D-regularity

(see Definition 3.17) for a certain set of directions D. Once again, these theorems require that trial points be filtered by the poll center (rather than a different filter point) infinitely often in the subsequence.

As is the case with the Filter GPS algorithm, convergence to a KKT point cannot be guaranteed with respect to the continuous domain, since there is no guarantee that the negative gradient lies inside the normal cone; however, we can specify a cone inside the tangent cone, whose polar contains the negative gradient.

> **Theorem 6.20** Let $\hat{p}$ be a limit point of a refining subsequence $\{p_k\}_{k \in K}$, and let V_d be the cone generated by all limit directions $d \in D$ of $\hat{p}$, for which $f(p_k) \leq f(p_k + \Delta_k d)$ holds infinitely often. Suppose that f satisfies the following conditions:
>
> 1. f is Lipschitz near $\hat{p}$ with respect to the continuous variables,
>
> 2. f is differentiable at $\hat{p}$ with respect to the continuous variables,
>
> 3. f is D-regular at $\hat{p}$ with respect to the continuous variables.
>
> Then under Assumptions A1–A4, $-\nabla^c f(\hat{p})$ belongs to the polar V_d° of V_d.

Proof. By Theorem 6.18, $f^\circ(\hat{p}; (d, 0)) \geq 0$ for all $d \in V_d$, and by Theorem 3.21, we have $\nabla^c f(\hat{p})^T w \geq 0$ for all $w \in V_d$. The result follows from the definition of a polar cone: $-\nabla^c f(\hat{p}) \in \{v \in \Re^n : v^T w \leq 0 \; \forall \; w \in V_d\}$. ∎

> **Theorem 6.21** Let $\hat{p}$, $\hat{y} \in \mathcal{N}(\hat{p})$, $\hat{z}$, and z_k be defined as in the statement of Theorem 6.8, with $\mathcal{N}$ continuous at $\hat{p}$, and suppose that $f(\hat{y}) < f(\hat{p}) + \xi$, where $\xi > 0$ is a lower bound on the extended poll triggers ξ_k^f and ξ_k^h for all k. Let V_d be the cone generated by all limit directions $d \in D$ of $\hat{z}$, for which $f(z_k) \leq f(z_k + \Delta_k d)$ holds infinitely often. Suppose that f satisfies the following conditions:

1. f is Lipschitz near $\hat{z}$ with respect to the continuous variables,

2. f is differentiable at $\hat{z}$ with respect to the continuous variables,

3. f is D-regular at $\hat{z}$ with respect to the continuous variables.

Then under Assumptions A1–A4, $-\nabla^c f(\hat{z})$ belongs to the polar V_d° of V_d.

Proof. By Theorem 6.19, $f^\circ(\hat{z};(d,0)) \geq 0$ for all $d \in V_d$, and by Theorem 3.21, we have $\nabla^c f(\hat{z})^T w \geq 0$ for all $w \in V_d$. The result follows from the definition of a polar cone: $-\nabla^c f(\hat{z}) \in \{v \in \Re^n : v^T w \leq 0 \ \forall \ w \in V_d\}$. ∎

As in Section 6.2.2, the remaining task is to characterize the limit point $\bar{y}$ of extended poll points $\{y_k^{j_k}\}_{k \in K}$ when $f(\hat{y}) = f(\hat{p})$. However, we cannot guarantee that $f(\bar{y}) = f(\hat{p})$ in general because, in the filter approach, we do not necessarily have $f(y_k^{j_k}) < f(y_k^{j_k-1})$ for all $j = 1, 2, \ldots, J_k$. In fact, a counterexample is found in [8]. Thus, to obtain the result of $f(\bar{y}) = f(\hat{p})$, we must add this as a hypothesis, as the following theorem shows.

Theorem 6.22 Let $\hat{p}$, $\hat{y} \in \mathcal{N}(\hat{p})$, and $\hat{z}$ be defined as in the statement of Theorem 6.8, with $\mathcal{N}$ continuous at $\hat{p}$, and suppose that the extended poll centers $\{y_k^j\}_{j=0}^{J_k}$ satisfy $f(y_k^j) \leq f(y_k^{j-1})$, $j = 1, 2, \ldots, J_k$ for infinitely many $k \in K$. Let $\bar{y} = (\bar{y}^c, \hat{y}^d)$ denote the limit point of the sequence of extended poll centers $\{y_k^{j_k}\}_{k \in K}$, where j_k is arbitrary. If f is continuous at $\hat{p}$ and $\hat{y}$ with respect to the continuous variables, then under Assumptions A1–A4, $f(\bar{y}) = f(\hat{p})$.

Moreover, if $\hat{y} \neq \hat{z}$, then there are infinitely many of these limit points, and the index J_k of the extended poll endpoint $y_k^{J_k}$ at iteration k satisfies $J_k \to +\infty$ (in K).

Proof. Let $\hat{y}$ in $\mathcal{N}(\hat{p})$ satisfy $f(\hat{y}) = f(\hat{p})$, and let $\bar{y}$ be a limit point of the sequence of extended poll points $\{y_k^{j_k}\}_{k \in K}$, distinct from $\hat{y}$ and $\hat{z}$.

Since, by assumption, $f(\hat{p}) \leq f(y_k^{j_k}) < f(y_k^{j-1})$, $j = 1, 2, \ldots, J_k$ for infinitely many $k \in K$ and $f(y_k^0)$ converges to $f(\hat{p})$, we conclude that $f(\bar{y}) = f(\hat{p})$. The remainder of the proof is exactly the same as the proof of Theorem 5.18. ∎

Remark 6.23 The additional hypothesis in Theorems 6.17–6.21 that the trial points be filtered by the current poll center infinitely often in the associated subsequence are automatically satisfied by either of the following two conditions:

1. The poll center (or extended poll center) is chosen to be the incumbent best feasible point (or best feasible point with respect to the local filter) infinitely often in the subsequence; *i.e.*, $p_k = p_k^F$ (or alternatively $z_k = z_k^F$) for infinitely many $k \in K$. To see this for p_k, observe that $p_k = p_k^F$ infinitely often means that $h(p_k) = 0$ for infinitely often, and since these p_k are mesh isolated poll centers, $f(p_k) \leq f(p_k + \Delta_k d)$ for all limit directions $d \in D$ for $\hat{p}$. Note that $p_k = p_k^F$ is chiefly an algorithmic choice, rather than a problem-dependent condition.

2. The limit point is strictly feasible with respect to the nonlinear constraints C, and C is continuous at the limit point. This holds because these two conditions ensure that for all sufficiently large $k \in K$, $p_k = p_k^F$. ∎

Remark 6.24 If there are no nonlinear constraints, then $p_k = p_k^F$ for all k. Thus, the results presented in this section encompass all of the results of Chapter 5.

Before presenting a stronger version of the algorithm, we must point out one other key result that we adapt from Theorem 4.21, whose proof is adapted from [7].

Theorem 6.25 If h and f are strictly differentiable at poll center p_k with respect to the continuous variables, and if $\nabla^c f(p_k) \neq 0$, then there cannot be infinitely many consecutive iterations where p_k is a mesh isolated poll center.

Proof. Let f and h be strictly differentiable at p_k with respect to the continuous variables, where $\nabla^c f(p_k) \neq 0$. Suppose that there are infinitely many iterations where p_k is a mesh isolated filter point. Let d be a direction associated with the (constant) subsequence of poll centers such that $\nabla^c f(p_k)^T d < 0$.

Since f is strictly differentiable at p_k with respect to the continuous variables, there exists an $\epsilon > 0$ such that either $h(p_k + \Delta(d, 0)) \leq h(p_k) < h_{\max}$, or $h(p_k + \Delta(d, 0)) > h(p_k)$, for all $0 < \Delta < \epsilon$.

If the first condition is satisfied, then for $\Delta_k < \epsilon$, the POLL step will find an unfiltered point, a contradiction. If the second condition is satisfied, then let $\tilde{h}$ be the smallest value of

$$\{h(x) : h(x) > h(p_k), x \in \mathcal{F}_k\} \cup \{h_{\max}\},$$

and let $\tilde{f}$ be the corresponding objective function value; *i.e.*, either $\tilde{f} = f(\tilde{x})$ for the vector $\tilde{x} \in \mathcal{F}_k$ that satisfies $h(\tilde{x}) = \tilde{h}$, or $\tilde{f} = -\infty$ in the case where $\tilde{h} = h_{\max}$. It follows that $\tilde{h} > h(p_k)$ and $\tilde{f} < f(p_k)$. Therefore, for sufficiently small $\Delta_k < \epsilon$, we have $h(p_k) < h(p_k + \Delta_k d) < \tilde{h}$ and $\tilde{f} < f(p_k + \Delta_k d) < f(p_k)$; thus, the trial mesh point is unfiltered, a contradiction. ∎

A limitation of this result is that, while it prevents a non-stationary p_k from being a mesh-isolated poll center for infinitely many consecutive iterations, it does

not completely prevent the algorithm from stalling there. The algorithm could still generate an infinite number of consecutive iterations in which p_k is either a mesh-isolated filter point or a filter point that does not generate a new poll center. If, for example, p_k simply alternates between these two possibilities, then Theorem 6.25 holds, but the algorithm still stalls at p_k.

As in previous results, the additional hypothesis of $p_k = p_k^F$ for infinitely many $k \in K$ would fully prevent stalling because it would force $h(p_k) = 0$ for infinitely many $k \in K$, and the strict differentiability of f at p_k means that $\nabla^c f(p_k)d < 0$ for some direction $d \in D_k$. Thus, for sufficiently large $k \in K$, Δ_k is sufficiently small to force $f(p_k + \Delta_k d) < f(p_k)$, and the algorithm moves to a new point.

6.2.4 Stronger Results

In this section, we present a modified version of the FMGPS algorithm, in which stronger results are gained by requiring the algorithm to evaluate every extended poll point, even if a point is found that is unfiltered with respect to the local filter. The price for the stronger result is additional computational cost.

Specifically, the STRONG EXTENDED POLL step involves the following two changes at each iteration k:

STRONG EXTENDED POLL step (at iteration k):

- For $j = 1, 2, \ldots, J_k$, the next poll center y_k^{j+1} of EXTENDED POLL step j is updated the local filter is updated with *all* unfiltered continuous neighbors of the current extended poll center y_k^j.

- The same set of directions $D_k(y_k) \subset D(y_k)$ must be used throughout the STRONG EXTENDED POLL step.

The two theorems below give the optimality results for h at $\bar{y}$ under different assumptions. The first assumes local Lipschitz continuity, while the second further assumes differentiability and D-regularity (see Definition 3.17).

Theorem 6.26 Let $\hat{p}$ and $\hat{y} \in \mathcal{N}(\hat{p})$ be defined as in the statement of Theorem 6.8, with $\mathcal{N}$ continuous at $\hat{p}$, and suppose $h(\hat{y}) = h(\hat{p})$. Let $\bar{y}$ be a limit point of strong extended poll points $\{y_k^{j_k}\}_{k \in K}$, where j_k is arbitrary, and let $d \in D$ be any limit direction of $\bar{y}$. Under Assumptions A1–A4, if h is Lipschitz near $\bar{y}$, then $h^\circ(\bar{y}; (d, 0)) \geq 0$.

Proof. Suppose by way of contradiction that there exists a $d \in D$ such that $h^\circ(\bar{y}; (d, 0)) < 0$. Then by Definition 3.9, we have

$$\limsup_{y \to \bar{y}, t \downarrow 0} \frac{1}{t} [h(y + t(d, 0)) - h(y)] < 0.$$

By local Lipschitz continuity, there exists an $\epsilon > 0$ such that $h(\bar{y} + t(d, 0)) < h(\bar{y})$ for all $t \in (0, \epsilon)$. Furthermore, without loss of generality, for each $t \in (0, \epsilon)$, there exists an $\epsilon_t > 0$ such that $h(\bar{y} + t(d, 0) + s(b, 0)) < h(\bar{y})$ for all $s \in (0, \epsilon_t)$ and for all $b \in \Re^n$ with $\|b\| = 1$.

Choose $k \in K$ sufficiently large such that $\|y_k^{j_k} - \bar{y}\| < \epsilon_t$ and $\Delta_k < \epsilon$. Now observe that $y_k^{j_k} = \bar{y} + \hat{s}(\hat{b}, 0)$ for some $\hat{s} \in \Re$ and $\hat{b} \in \Re^n$ with $\|\hat{b}\| = 1$. Then $\hat{s} = \hat{s}\|(\hat{b}, 0)\| = \|y_k^{j_k} - \bar{y}\| < \epsilon_t$. Use of the STRONG EXTENDED POLL step (with $j_k < J_k$) guarantees that $h(y_k^{j_k+1}) \leq h(y_k^{j_k} + \Delta_k(d, 0)) = h(\bar{y} + \hat{s}(b, 0) + \Delta_k(d, 0)) = h(\bar{y} + \Delta_k(d, 0) + \hat{s}(b, 0)) < h(\bar{y})$, since Δ_k is a specific case of $t \in (0, \epsilon)$ and $\hat{s} \in (0, \epsilon_t)$. Thus, $h(y_k^{j_k+1}) < h(\bar{y}) = h(\hat{p}) \leq h(p_k)$, which contradicts the assumption that p_k is a mesh isolated poll center. ∎

Theorem 6.27 Let the assumptions of Theorem 6.26 hold, and suppose that h satisfies the following conditions:

1. h is Lipschitz near $\bar{y}$ with respect to the continuous variables,

2. h is differentiable at $\bar{y}$ with respect to the continuous variables,

3. h is D-regular at $\bar{y}$ with respect to the continuous variables.

Then under Assumptions A1–A4, $\nabla^c h(\bar{y})^T w \geq 0$ for all $w \in T_{X^c}(\bar{y})$. If $X^c = \Re^{n^c}$ or if $\bar{y}$ lies in the interior of X^c, then $0 = \nabla^c h(\bar{y}) \in \partial^c h(\bar{y})$.

Proof. Since Theorem 6.26 applies to a set of directions that form a positive spanning set for $T_{X^c}(\bar{y})$, the result follows directly from it and Theorem 3.21. ∎

Because of the construction of the filter, equivalent results for the objective function f require stronger assumptions, similar to those in the results of Section 6.2.3.

Theorem 6.28 Let $\hat{p}$ and $\hat{y} \in \mathcal{N}(\hat{p})$ be defined as in the statement of Theorem 6.8, with $\mathcal{N}$ continuous at $\hat{p}$, and assume that $f(\hat{y}) = f(\hat{p})$. Let $\bar{y}$ be a limit point of strong extended poll centers $\{y_k^{j_k}\}_{k \in K}$, where j_k is arbitrary, and let $d \in D$ be any limit direction of $\bar{y}$. Under Assumptions A1–A4, if f is Lipschitz near $\bar{y}$ with respect to the continuous variables, and $f(y_k^{j_k}) < f(y_k^{j_k - 1})$, $j_k = 1, 2, \ldots, J_k$, holds for infinitely many $k \in K$, then $f^{\circ}(\bar{y}; (d, 0)) \geq 0$.

Proof. Suppose by way of contradiction that there exists a $d \in D$ such that $f^{\circ}(\bar{y}; (d, 0)) < 0$. Then by Definition 3.9, we have

$$\limsup_{y \to \bar{y}, t \downarrow 0} \frac{1}{t} \left[f(y + t(d, 0)) - f(y) \right] < 0.$$

By local Lipschitz continuity, there exists an $\epsilon > 0$ such that $f(\bar{y} + t(d, 0)) < f(\bar{y})$ for all $t \in (0, \epsilon)$. Furthermore, without loss of generality, for each $t \in (0, \epsilon)$, there exists an $\epsilon_t > 0$ such that $f(\bar{y} + t(d, 0) + s(b, 0)) < f(\bar{y})$ for all $s \in (0, \epsilon_t)$ and for all $b \in \Re^n$ with $\|b\| = 1$.

Choose $k \in K$ sufficiently large such that $\|y_k^{j_k} - \bar{y}\| < \epsilon_t$ and $\Delta_k < \epsilon$. Now observe that $y_k^{j_k} = \bar{y} + \hat{s}(\hat{b}, 0)$ for some $\hat{s} \in \Re$ and $\hat{b} \in \Re^n$ with $\|\hat{b}\| = 1$. Then $\hat{s} = \hat{s}\|(\hat{b}, 0)\| = \|y_k^{j_k} - \bar{y}\| < \epsilon_t$.

Since, by assumption, we have $f(y_k^{j_k}) < f(y_k^{j_k-1})$, $j = 1, 2, \ldots, J_k$ for infinitely many $k \in K$, use of the STRONG EXTENDED POLL step (with $j_k < J_k$) guarantees that $f(y_k^{j_k+1}) < f(y_k^{j_k} + \Delta_k(d, 0)) = f(\bar{y} + \hat{s}(b, 0) + \Delta_k(d, 0)) = f(\bar{y} + \Delta_k(d, 0) + \hat{s}(b, 0)) < f(\bar{y})$ for infinitely many $k \in K$, since Δ_k is a specific case of $t \in (0, \epsilon)$ and $\hat{s} \in (0, \epsilon_t)$. Thus, $f(y_k^{j_k+1}) < f(\bar{y}) = f(\hat{p}) \leq f(p_k)$, which contradicts the assumption that p_k is a mesh isolated poll center. ∎

Theorem 6.29 Let the assumptions of Theorem 6.28 hold, and let V_d be the cone generated by all limit directions $d \in D$ of $\bar{y}$ for which $f(y_k^{j_k}) \leq f(y_k^{j_k} + \Delta_k d)$, $j_k = 0, 1, \ldots, J_k$, holds for infinitely many $k \in K$. If f satisfies the assumptions of Theorem 6.20 for $\bar{y}$, then under Assumptions A1–A4, $-\nabla^c f(\bar{y})$ belongs to the polar V_d° of V_d.

Proof. By Theorem 6.28, $f^\circ(\bar{y}; (d, 0)) \geq 0$ for all $d \in V_d$, and by Theorem 3.21, we have $\nabla^c f(\bar{y})^T w \geq 0$ for all $w \in V_d$. The result follows from the definition of a polar cone: $-\nabla^c f(\bar{y}) \in \{v \in \Re^n : v^T w \leq 0 \ \forall \ w \in V_d\}$. ∎

6.3 Summary

The FMGPS Algorithm presented and analyzed here represents a generalization of much of the work of Torczon [130], Lewis and Torczon [94, 95], and Audet and Dennis [6, 7, 8]. That is, the algorithms described in these papers are specific instances of the FMGPS class of algorithms, and the convergence theory presented here applies to a broader class of problems that includes the work in these papers.

In summary, if the Assumptions A1-A4 are satisfied, then we get the following hierarchy of convergence results:

- If h is lower semi-continuous at any limit point $\bar{p}$ of the GPS iteration sequence, then $h(\bar{p}) \leq \lim_k h(p_k)$. A similar result for f requires that $p_k = p_k^F$ for all but finitely many k.

- Every limit point of the iteration sequence at which h is continuous has the same function value $\lim_k h(p_k)$, whether or not it is a stationary point. A similar result for f requires that $p_k = p_k^F$ for all but finitely many k.

- Theorem 6.8 defines and establishes the existence of the limit points $\hat{p}$, $\hat{y}$, and $\hat{z}$, a limit point of extended poll endpoints.

- If h is lower semicontinuous at $\hat{p}$ with respect to the continuous variables and continuous at $\hat{y}$ with respect to the continuous variables, and if $\mathcal{N}$ is continuous at $\hat{p}$, then $h(\hat{p}) \leq h(\hat{y})$. A similar result for f requires that y_k be filtered by the current poll center infinitely often in the subsequence.

- If the function h is Lipschitz near $\hat{p}$ with respect to the continuous variables, then $h^\circ(\hat{p}; (d, 0)) \geq 0$ for any limit direction d of $\hat{p}$. A similar result holds at $\hat{z}$. Similar results for f at $\hat{p}$ and $\hat{z}$ require that $p_k + \Delta_k d$ (or $z_k + \Delta_k d$) be filtered by the current poll center infinitely often in the subsequence.

- If h is strictly differentiable (or its equivalent on the tangent cone) at $\hat{p}$ with respect to the continuous variables, then $\hat{p}$ is a first order stationary point with respect to the continuous variables. A similar result holds at certain $\hat{z}$.

- If f is strictly differentiable (or its equivalent on the tangent cone) at $\hat{p}$ with respect to the continuous variables, then the polar of the cone formed by certain

limit directions of $\hat{p}$ contains $-\nabla f(\hat{p})$. While a first order stationary cannot be guaranteed in general, it can be if $\hat{p}$ is strictly feasible. A similar result holds at certain $\hat{z}$.

- If a stronger version of the algorithm is used, in which the entire extended poll set is evaluated at every step, then the preceding three items apply to all limit points of extended poll points associated with refining subsequences, and not just the endpoints of EXTENDED POLL steps. In essence, this allows for a stronger optimality condition in a special case.

In Chapter 7, the FMGPS algorithm is applied to the problem of designing a load bearing thermal insulation system.

Chapter 7

Designing an Optimal Thermal Insulation System

In this chapter, a new Matlab® implementation of the FMGPS algorithm is described, and is then applied to the design of a load-bearing thermal insulation system. The goal of the optimization problem is to minimize the power required to properly operate the system, subject to various bound, linear, and nonlinear constraints. Computational results demonstrate that the algorithm performs in accordance with the theoretical results described in Chapter 6.

7.1   NOMADm Software

The FMGPS algorithm has been coded in Matlab® function form and incorporated into an interactive Matlab® software package, called NOMADm, available for download on the internet [1]. The NOMADm software requires the user to supply up to five Matlab® function files as follows:

1. a function defining the objective and nonlinear constraint functions (and optionally any available derivative information),

2. a function that returns the initial iterate,

3. a function that returns the bound vectors and coefficient matrix that define the bound and linear constraints (if these exist),

4. a function defining the discrete set of neighbors (required only for MVP problems), and

5. a function that defines any parameters used by the other user files (optional).

Additionally, NOMADm has the following features:

- An implementation of the algorithm given in Figure 4.3 scheme for computing tangent cone generators of a linearly constrained region, so that polling directions always conform to the geometry of the constraints (see Definition 4.3).

- A cache or database of previously computed iterates so that, in the case of expensive function evaluations, multiple evaluations of the same point are avoided.

- Several common choices for the SEARCH step, including additional polls around other filter points and Latin hypercube search (see [107, 126]). It also includes "hooks" for user-defined search and surrogate functions, so that users can seamlessly attach their own favorite method to the code.

- Optional scaling of the mesh directions in a manner similar to that of [41].

- Additional optional termination criteria, including limits on the number of GPS iterations, number of function calls, and CPU time.

- Real-time plots of the filter (similar to Figure 4.5) and iteration history.

- Options for using derivative information (see Chapter 8) to reduce the number of function evaluations required for convergence.

7.2   Problem Description

In a thermal insulation system, heat intercepts are often used to minimize the heat flow from a hot to a cold surface (or *vice versa*). Figure 7.1 illustrates an example of such a system of fixed length L, in which power is applied to maintain each intercept i at a cooling temperature $\overline{T}_i, i = 1, 2, \ldots, n$. An insulator of thickness x_i is placed between each pair of intercepts $i-1$ and i, with the convention that $i = 0$ and $i = n+1$

represent the cold and hot surfaces, respectively, so that $\overline{T}_0 = \overline{T}_C$ and $\overline{T}_{n+1} = \overline{T}_H$. Note that each insulator in Figure 7.1 may have a different cross-sectional area. The design variables for the system include the number and cooling temperatures of the intercepts, and the insulator types and thicknesses. Furthermore, we assume that the system must be load-bearing, meaning that the insulators act as mechanical supports; thus, only solid materials can be used.
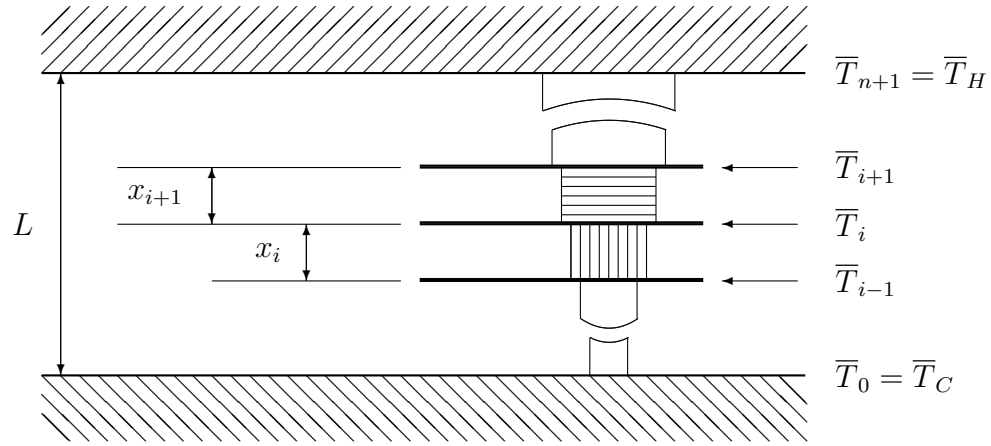
Figure 7.1   Schematic of a Thermal Insulation System

Variations of this type of system used in cryogenic engineering applications, such as superconducting magnetic energy storage systems and space borne magnets, have been studied by several authors. Hilal and Boom [71] used a gradient-based optimizer to minimize power for $n = 1$, 2, and 3 intercepts with two choices of insulators of constant cross-sectional area, but without mixing insulator types within the system. Hilal and Eyssa [72] studied the same problem, but with variable cross sectional area for the mechanical supports. In considering systems with more general types of insulation, Chato and Khodadidi [30] sought to minimize entropy, similar to the formulation given by Bejan [15]. Other related attempts to optimize the design of

these systems are found in [99], [112], and [137]. An actual implementation of these types of systems for the Large Hadron Collider (LHC) project is discussed in three technical reports [45, 81, 105]. While all of these studies vary in geometry and fidelity of the underlying models, none of them optimizes with respect to the categorical variables; namely, the number of intercepts and types of insulators.

Kokkolaras, Audet, and Dennis [89] extended the Hilal and Boom model to include the categorical variables in the optimization problem, allowing for a mixture of different types of insulators in the system. They achieved a 65% reduction in required refrigeration power from that of [71] using the MGPS algorithm described in Chapter 5 to solve this bound constrained MVP problem. However, other than the selecting the specific list of possible insulators, they did not consider certain load-bearing aspects of the problem, such as thermal expansion, system mass, and stress.

7.2.1 Basic Model Formulation

In this section, we reformulate the Kokkolaras *et al.* model (as extended from that of [71]) to include constraints on thermal expansion, stress, and mass. Much of this discussion comes directly from the presentation in [89], although some of the discussion of stress is similar to that of [72]. The new model can be expressed as

$$\min_{(n,I,x,\overline{T}) \in X} \quad f(n, I, x, \overline{T})$$
$$\text{s. t.} \quad g(x) \leq 0, \tag{7.1}$$

with the following nomenclature:

- n is the number of heat intercepts, with the convention that the cold and hot walls are numbered 0 and $n + 1$, respectively.

- $I \in \mathcal{I}^{n+1}$ is the set of insulators used, where I_i represent the insulator type between intercepts $i-1$ and i, and $\mathcal{I}$ denotes the finite set of possible insulator types.

- $x \in \Re^n$ is the vector whose i-th component is the thickness of the i-th insulator, with the convention that $x_{n+1} = L - \sum_{i=1}^{n} x_i$.

- $\overline{T} \in \Re^n$ is the vector whose i-th component is the temperature of the i-th intercept, with the convention that $\overline{T}_0 = \overline{T}_C$ and $\overline{T}_{n+1} = \overline{T}_H$.

- The feasible region X is defined by the following linear and categorical constraints:

$$n \in \{1, 2, \ldots, n_{\max}\}, \tag{7.2}$$

$$I \in \mathcal{I}^{n+1}, \tag{7.3}$$

$$\sum_{i=1}^{n} x_i \leq L, \tag{7.4}$$

$$x_i \geq 0, \quad i = 1, \ldots, n \tag{7.5}$$

$$\overline{T}_{i-1} \leq \overline{T}_i \leq \overline{T}_{i+1}, \quad i = 1, \ldots, n. \tag{7.6}$$

A difficulty in solving this problem is that the dimension of the vectors I, x, and $\overline{T}$ depend on the variable n. For any value of n, there are $n+1$ other categorical variables and $2n$ continuous variables, yielding a total of $3n+2$ variables.

7.2.2 Objective Function

The objective function represents the total refrigeration power of the system; thus,

$$f = \sum_{i=1}^{n} P_i, \tag{7.7}$$

where P_i is the power applied to intercept i, $i = 1, 2, \ldots, n$. The required power to keep intercept i at a fixed temperature $\overline{T}_i$ is given by

$$P_i = C_i \left(\frac{\overline{T}_H}{\overline{T}_i} - 1 \right) (q_{i+1} - q_i), \qquad (7.8)$$

where C_i (a function of temperature) is the thermodynamic cycle efficiency coefficient at intercept i, and q_i represents the heat flow from intercept i to $i - 1$.

In general, heat flow q through a volume is governed by Fourier's law,

$$q dx = A k(T) dT, \qquad (7.9)$$

where A (a function of the spatial coordinates in the z-y plane perpendicular to the x coordinate) is the cross-sectional area of the volume, and k (a function of the temperature variable T) is the effective thermal conductivity of the volume.

For the problem at hand, the heat flow q_i from intercept i to $i - 1$ is given by

$$q_i = \frac{A_i}{x_i} \int_{\overline{T}_{i-1}}^{\overline{T}_i} k(T; I_i) dT, \quad i = 1, 2, \ldots, n + 1, \qquad (7.10)$$

where A_i denotes the cross-sectional area of insulator i, and the thermal conductivity k is a function of both the temperature T and the type of insulator I_i.

By substituting (7.10) into (7.8), the objective function can be expressed by (7.7), with

$$P_i = C_i \left(\frac{\overline{T}_H}{\overline{T}_i} - 1 \right) \left[\frac{A_{i+1}}{x_i} \int_{\overline{T}_i}^{\overline{T}_{i+1}} k(T; I_i) dT - \frac{A_i}{x_{i-1}} \int_{\overline{T}_{i-1}}^{\overline{T}_i} k(T; I_i) dT \right]. \qquad (7.11)$$

7.2.3 Nonlinear Constraints

The inequality in (7.1) represents the new nonlinear constraints added to the MVP problem of [89], which allow us to numerically test the FMGPS algorithm. Although we add three types of constraints; namely mass, stress, and thermal expansion, we

model the stress constraint as an implicit equality constraint so that we can eliminate insulator cross-sectional areas as design variables.

The first constraint concerns the overall mass of the system. This constraint may actually be budgetary in nature, as larger amounts of insulation material increase the overall cost. Since the weight of each insulator can be expressed as a product of the density of the insulator material and its volume, we have the constraint,

$$\sum_{i=1}^{n} \rho_i(I_i) A_i x_i \leq m_{\text{max}}, \tag{7.12}$$

where $\rho_i(I_i)$ represents the density of material used as insulator i, and m_{max} is the maximum allowable mass of the system. Note that setting each volume to $A_i x_i$ assumes that the area of the insulator is constant throughout the interval; however, this is not an unreasonable assumption.

We also assume that the system must be capable of bearing a specified load (or force) F. The stress, defined as load per unit area, must not be allowed to exceed a certain level. In this case, we constrain the stress applied to each insulator to be no greater than the tensile yield strength of that insulator (thus, we assume that the load is suspended from the system, rather than resting on top of it). The tensile yield strength at insulator i, denoted by $\sigma_i(T; I_i)$, is a function of both the insulator type and temperature. Thus, we have constraints of the form

$$\frac{F}{A_i} \leq \bar{\sigma}_i \doteq \min\left\{ \sigma_i(T; I_i) : \overline{T}_{i-1} \leq T \leq \overline{T}_i \right\}, \quad i = 1, 2, \ldots, n+1. \tag{7.13}$$

The difficulty with the constraints given in (7.12) and (7.13)) is that they treat the areas A_i, $i = 1, \ldots, n$, as additional design variables. However, there is a convenient and perfectly legitimate way around this problem. First, observe that the direct relationship between A_i and P_i in (7.11) means that decreasing the cross-sectional area of any insulator reduces the power applied to the corresponding intercept, which

is exactly the goal of the optimization problem. Thus, at an optimal point, each A_i should be as small as possible. Furthermore, as each A_i is made smaller, the stress constraint in (7.13) becomes binding – and must be so at optimality. Therefore, we can assume that (7.13) holds with equality, solve for A_i, and substitute the resulting equation into (7.12). We must also make this substitution into the objective function. This eliminates A_i as a design variable and yields a stress-mass constraint of the form

$$\sum_{i=1}^{n} \rho_i(I_i)\frac{x_i}{\bar{\sigma}_i} \leq \frac{m_{\max}}{F}. \tag{7.14}$$

The final constraint is one associated with the system's thermal expansion, or in our cryogenic application, thermal contraction. In addition to moving the heat intercepts out of their optimal position, thermal contraction causes additional stress on the materials and, if excessive, can cause other difficulties, such as deformations in the material. The development of this constraint presented here is adapted from [13].

Since different insulators at different temperatures exhibit different contraction behaviors, we must treat thermal contraction of each insulator separately as a change in its thickness. The constraint can then be expressed as a weighted sum, where each insulator's weight is simply its thickness divided by the total length of the system; *i.e.*,

$$\sum_{i=1}^{n} \left(\frac{\Delta x_i}{x_i}\right)\left(\frac{x_i}{L}\right) \leq \frac{\delta}{100}, \tag{7.15}$$

where δ is a limit on the percent total contraction of the system.

For insulator i, $e(T; I_i)$ denotes the unit thermal contraction from intercept i to any point between intercept i and $i-1$ (a function of temperature) and is computed by the formula,

$$e(T; I_i) = \int_{T}^{\overline{T}_i} \lambda dT, \tag{7.16}$$

where λ is the linear coefficient of thermal expansion for the insulator type. Precomputed values for $e(T; I_i)$ are available in lookup tables for a wide range of temperatures and for several material types [13, 108]. Total thermal contraction for an insulator i is given by

$$\frac{\Delta x_i}{x_i} = \frac{\int_{\overline{T}_{i-1}}^{\overline{T}_i} e(T; I_i) k(T; I_i) dT}{\int_{\overline{T}_{i-1}}^{\overline{T}_i} k(T; I_i) dT}; \tag{7.17}$$

thus, the nonlinear thermal expansion constraint is given by

$$\sum_{i=1}^{n} \left[\frac{\int_{\overline{T}_{i-1}}^{\overline{T}_i} e(T; I_i) k(T; I_i) dT}{\int_{\overline{T}_{i-1}}^{\overline{T}_i} k(T; I_i) dT} \right] \left(\frac{x_i}{L} \right) \leq \frac{\delta}{100}. \tag{7.18}$$

The resulting optimization problem can now be expressed by the objective function in (7.7) and (7.11), the linear constraints defined in (7.2)–(7.6), and the nonlinear constraints defined by (7.14) and (7.18).

7.3 Computational Model

This section further describes the optimization problem in terms of modelling decisions, material data sources, and other problem setup issues.

7.3.1 Material Data

The types of insulators were chosen as the same as in [89]; namely, nylon, teflon, fiberglass epoxy (both normal and plane), 6063-T5 aluminum, 1020 low-carbon steel, and 304 stainless steel. For each of these materials, a substantial amount of engineering data was required. Thermal conductivity and contraction data were obtained from lookup tables in [13] and [108], while material densities were found in [125] and [108], and tensile yield strength data were obtained from [44] and [106].

Thermodynamic cycle efficiency coefficients C_i, $i = 1, 2, \ldots, n$ (see (7.8)) are dependent on temperature as follows:

$$C_i = \begin{cases} 5, & \text{if } T < 4.2K \\ 4, & \text{if } 4.2K \leq T < 71K \\ 2.5, & \text{if } T \geq 71K. \end{cases} \tag{7.19}$$

In order to make the model as accurate and efficient as possible, cubic splines were used to fit all of the data found in lookup tables, including thermal conductivity, thermal contraction, and tensile strength data. Numerical integrations were performed by applying a composite Simpson's Rule, with nodes matching those of the cubic spline. This eliminates truncation error, since Simpson's Rule is exact for cubic polynomials [85].

7.3.2 Choosing Discrete Neighbors

Recall from the discussion in Section 5.1, and especially from Definition 5.3, that the neighborhood structure that the user chooses to incorporate determines the definition of a minimizer. That is, when the solution of an MVP problem is found, it is with respect to the user-specified discrete set of neighbors. If a neighborhood structure is chosen so that all other sets of discrete variable values are neighbors, a more global solution can be obtained, but at extraordinary computational cost. On the other hand, severely restricting the size of the set of neighbors will save significant computational cost, but a local optimizer with a higher objective function value is likely to be obtained.

In order to make proper comparisons, the set of neighbors we use for this problem is exactly the same as was used by Kokkolaras *et al.* [89]. It includes designs in which the following occur:

- The type of insulator between any two heat intercepts is changed to any other type, while insulator thicknesses and intercept temperatures remain constant.

- An intercept and the insulator above it are removed, while thicknesses of the remaining insulators are increased proportionally (rounded to the nearest integer multiple of the current mesh size) to fill the remaining space.

- A new intercept and an insulator underneath it are added with the following properties:

  - The type of insulator is the same as the one below it,

  - The cooling temperature is set to the average of the two intercepts adjacent to it, rounded to the nearest integer multiple of the current mesh size,

  - The thickness of both the new insulator and the insulator below it are both set to half of that of latter, rounded to the nearest integer multiple of the current mesh size.

Note that rounding to the nearest integer multiple of the current mesh size is necessary to ensure that the trial point lies on the mesh.

7.4 Computational Results

We now present results of several NOMADm runs and show that the approach significantly improves the design of Hilal and Eyssa [72], and is comparable to the results of Kokkolaras *et al.* [89], even though the problem takes on the additional nonlinear load-bearing constraints.

Table 7.1 shows the data parameters chosen for the results that follow. The first four have values identical to the choices of Kokkolaras *et al.* [89], while the remaining parameters are unique to this problem.

Table 7.1   Thermal Insulation Problem: Model Parameters

Parameter	Symbol	Value
Hot surface temperature	$\overline{T}_H$	300 K
Cold surface temperature	$\overline{T}_C$	4.2 K
System total length	L	100 cm
Maximum number of intercepts	n_{max}	10
Load placed on the system	F	250 kN
Maximum total system mass	m_{max}	10 kg
Maximum system thermal contraction	δ	5%

To match the setup of [89] as much as possible, runs were performed with an initial mesh size of $\Delta_0 = 10$ and terminated when the condition $\Delta_k \leq .15625$ was achieved. Also, mesh coarsening was not used. The mesh refinement strategy used by [89], could not be duplicated by NOMADm; thus, in our runs, we refine by simply divide the mesh size parameter Δ_k in half. Extended poll triggers for the objective and constraint violation function were set at one and five percent, respectively, the former being consistent with [89].

Also to match the setup of [89], we set the initial design to consist of 1 intercept placed exactly in the middle of the system and set at 150 K, with a nylon insulator on the cold side and a teflon insulator on the hot side.

When the filter logic of the FMGPS is applied for the nonlinear constraints, the SEARCH step consists of a poll around the least infeasible point, while polling is performed around the best feasible point.

7.4.1   Validation

The software and function files were validated by mimicking the designs of [71] and [72], running these problems, and comparing the designs. In both of these previous

papers, the authors applied their optimizer to cases of 1-3 heat intercepts with insulators made of either 304 stainless steel or plane-cloth fiberglass epoxy. For stainless steel, our results matched theirs almost exactly. For fiberglass epoxy, there were some slight differences when more than one intercept is used, but these were noted in [89] as well. These differences were most likely caused by different methods in computing the objective function integrals. As in [89], we used cubic splines to fit thermal conductivity data that was available only in tabular form for specific temperatures. However, rather than apply a Matlab® integration routine, we applied our own implementation of Simpson's rule, which is exact for cubic polynomials. The inaccuracy is more visible in the epoxy results because thermal conductivity data was only available at four temperatures, as opposed to the 18 different temperatures available for stainless steel.

When we recomputed the results of [89] to validate our mixed variable logic, we converged to a different design with a similar low objective function value. Table 7.2 shows the differences between the two runs, where the materials cited there are abbreviated by the following: N = nylon, E = epoxy (normal), E_p = epoxy (plane), and T = teflon. Note that, although power is optimized in our software, we report normalized power at termination, in which power is multiplied by the system length L and divided by the smallest cross-sectional area of any insulator. Previous authors have expressed results this way so that designs can be compared, independent of these two parameters. We keep this convention for the same purpose.

While the numerical integration issue just described can lead to small deviations, we suspect that the difference in mesh refinement strategies led to a different local optimizer along a different path. In [89], with a starting mesh size of $\Delta_0 = 10$, the mesh refinement strategy was to divide the current mesh size by 2^ℓ, where $\ell = 1, 2, \ldots$ is incremented each time the mesh is refined. Since NOMADm is currently incapable

of incrementing ℓ, our mesh refinement consists of simply dividing the current mesh size by two (*i.e.*, $\ell = 1$).

Table 7.2 Thermal Insulation Problem: MVP Validation

Problem:	Original		FMGPS ReRun	
Power $\left(\frac{PL}{A}\right)$:	25.294 W/cm		25.589 W/cm	
Insulators Used:	$NNNNNNNEEET$		$NNNNNNTEETT$	
i	x_i (cm)	$\overline{T}_i$ (K)	x_i (cm)	$\overline{T}_i$ (K)
1	0.3125	4.2188	4.5313	6.125
2	5.4688	7.3438	6.7188	10.55
3	3.9062	10	4.8437	14.35
4	6.5625	15	4.2188	17.994
5	5.7812	20	7.3438	24.969
6	5.1562	25	9.8438	36.006
7	13.2812	40	24.948	71.094
8	21.4062	71.0938	12.135	116.88
9	8.5938	101.25	7.5	156.88
10	9.2188	146.25	6.4063	198.44
11	20.3125		11.5105	

7.4.2 Adding the Nonlinear Constraints

The nonlinear constraints were added to the runs in three steps. First, we simply tested the two designs from Table 7.2 versus the new nonlinear constraints and found that the thermal contraction constraint for both designs was violated by approximately 8%. This suggests that a new design having a different material configuration should be expected as the new constraints are incorporated. Second, the implicit constraint on stress (given by (7.13) with equality), was added to allow variable cross-sectional areas, and thus match the formulation of Hilal and Eyssa [72]. We re-

fer to this as the *partial model*. Finally, the mass and thermal contraction constraints were added to complete the full model. By doing so, the resulting change in required power represents the cost of satisfying the additional load-bearing constraints.

Table 7.3 shows the results for the partial and full models (columns 3 and 4), along with the design found by Hilal and Eyssa [72] (column 2). For each run, the minimal required power is normalized by multiplying by the total system length and dividing by the smallest cross-sectional area of an insulator. Following the power and the material configuration for each design, insulator thicknesses and heat intercept temperature settings are listed.

Table 7.3   Thermal Insulation Problem: Results

Problem:	Hilal & Eyssa		Partial Model		Full Model	
Power $\left(\frac{PL}{A}\right)$:	53.2 W/cm		24.551 W/cm		23.768 W/cm	
Insulators Used:	$E_pE_pE_p$		$EEEEEEEEEEE$		$EEEEEEEEEEE$	
i	x_i (cm)	$\overline{T}_i$ (K)	x_i (cm)	$\overline{T}_i$ (K)	x_i (cm)	$\overline{\overline{T}}_i$ (K)
1	22.0	8.38	7.1875	6.5875	0.625	4.25
2	23.8	36.3	11.406	12.938	8.125	7.7375
3	24.8	116.6	15.625	25.85	7.9688	12.369
4	29.4		29.531	71.094	7.8125	18.094
5			6.875	100.31	12.344	29.912
6			5	127.66	26.094	71.094
7			2.5	143.13	8.125	105.94
8			2.5	159.06	5.3125	135.47
9			4.6875	188.59	5	165.94
10			5	222.5	5.625	202.03
11			9.688		12.9682	

We can see immediately that the addition of the implicit stress constraint results in a variable cross-sectional area design that requires over 50% less (normalized) power

than the design found by Hilal and Eyssa [72]. This savings was expected because the newer formulation allows for varying the number of heat intercepts and the mixing of insulator types. A similar (65%) savings was achieved by Kokkolaras *et al.* [89] in optimizing the constant cross-sectional area formulation of Hilal and Boom [71].

A bit surprising was the slight decrease in power when the final two constraints are added (see columns 3 and 4), particularly because this is the point at which a linearly constrained problem becomes a nonlinearly constrained one, and the new algorithm's filter logic is applied. Recall that the theory ensures convergence to a first-order stationary point for the partial model (since all its constraints are linear), but does not do so for the full model. In spite of this, the new algorithm still found a better feasible design. It is indeed possible that the FMGPS algorithm generated a different sequence and simply terminated near a better local minimizer.

Figure 7.2 illustrates the performance of the FMGPS algorithm on the full model, where the power required for the incumbent best design is plotted versus the number of function evaluations. The lower plot is a magnification of upper one. The "L"-shaped plot is very typical behavior of derivative-free methods, since good stopping rules for these methods are difficult. The "stair steps" seen in the right-hand plot indicate varying length polling sequences.

We should note that the power values shown on the vertical axes of these plots do not match the data in the Table 7.3 because they represent two different things. The objective function is to minimize power, as measured in Figure 7.2, but the required power shown in Table 7.3 is normalized (hence the $\left(\frac{PL}{A}\right)$ notation), so as to allow comparisons with the results of Hilal and Eyssa [72].

Figure 7.3 depicts the progression of the filter during the run of the full model, where the plots in the right column are magnifications of those on the left. Each of the three rows represents a "snapshot" taken after 150, 200, and 500 respective
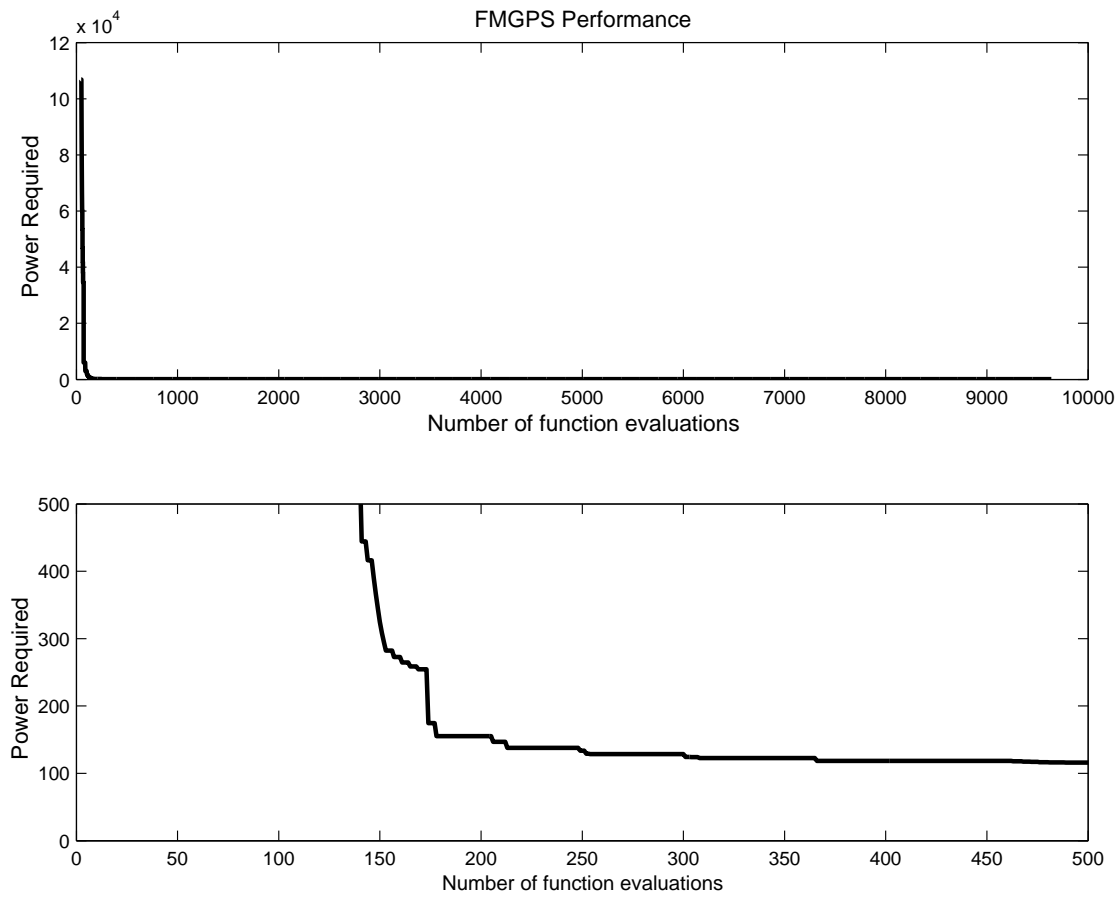
Figure 7.2   Iteration History for the
Thermal Insulation System Design Problem

function evaluations were performed. Although the algorithm terminated after more than 9000 function evaluations, changes in the filter after 500 function evaluations could not be detected within the resolution of the plot. This is consistent with the long and shallow progression of the best objective function value seen in Figure 7.2.

In the filter plots, the asterisks represent a subset of the best feasible points found up to that point, while the filter is represented by the "stair step" lines. In this run, the constraints were scaled by dividing each constraint by its right-hand side and

then subtracting one from both sides. Thus in the left column plots, the choice of $h_{\max} = 1$ represents a 100% constraint violation.
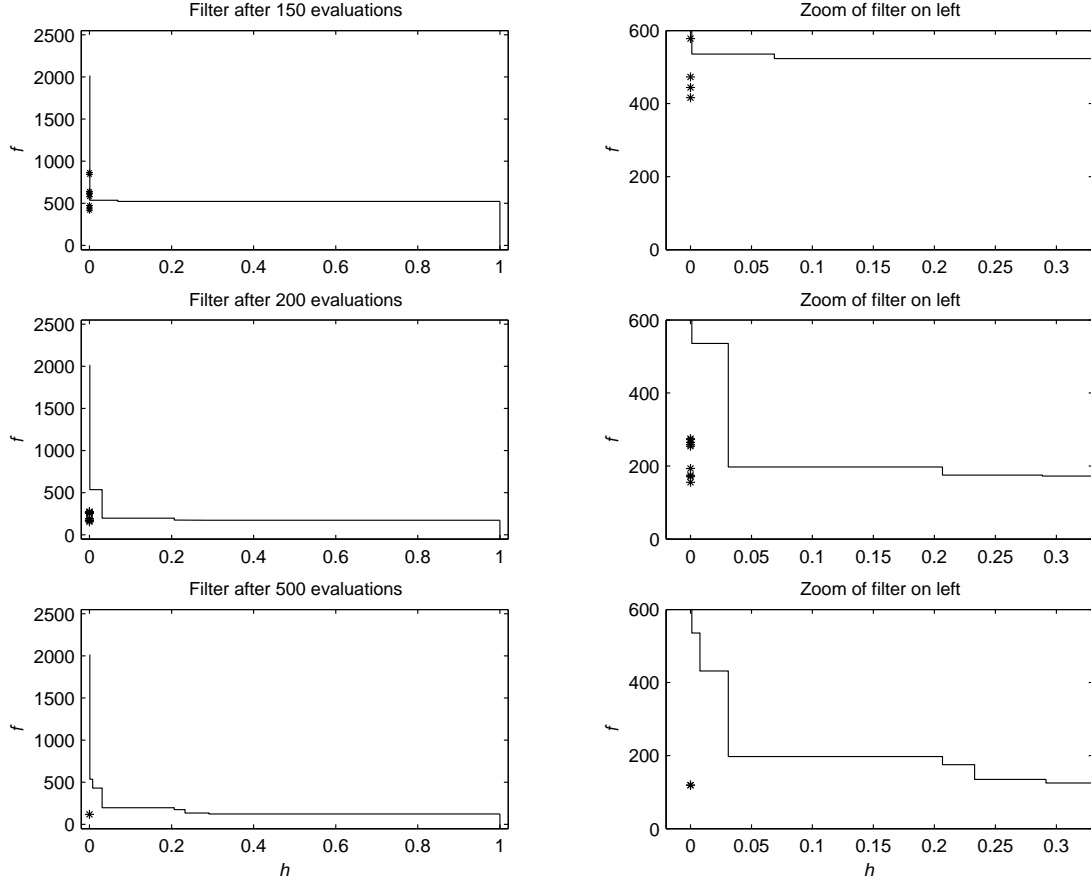
Figure 7.3   Filter Progression for the Full Model

Table 7.3 also shows that the new constraints yield significantly different insulator configurations than that of Kokkolaras *et al.*[89] – that of all normal cloth fiberglass epoxy. This is consistent with the raw data we used, which shows epoxy to have low thermal conductivity and higher resistance to stress than nylon or teflon.

However, some of the other materials (including nylon and teflon) have better thermal contraction properties than epoxy, which also has a low thermal stress threshold

and which would be tested by any thermal contraction. Modelling thermal stress as an additional constraint that depends on thermal contraction would be an interesting extension to this problem and might result in a completely different material configuration.

7.5 Conclusions

The results shown here demonstrate that, although the FMGPS algorithm can be expensive – particularly for mixed variable problems, it successfully generated much-improved designs for this problem. The design of [72] has been significantly improved, and the addition of constraints on stress, mass, and thermal contraction yields a more realistic feasible design with essentially no additional power required over that of [89].

One other extension of this problem that would be interesting to test in future work is to allow the cross-sectional area of each insulator to vary continuously within its interval. This requires some refinement of the mathematical model and the resulting computer code, but it is certainly achievable. We would also like to perform more runs that allow for more heat intercepts, so as to compare further results with [89], but NOMADm is not currently capable of performing the requisite number of function evaluations in a reasonable amount of time.

Chapter 8

GPS Algorithms with Derivative Information

8.1 Motivation

In this chapter, a key question is answered; namely, how to incorporate any available derivative information into GPS algorithms in such a way that can speed convergence, yet not sacrifice the enhanced global performance that can be implemented into these methods. The new material presented here is in collaboration with Audet and Dennis, and except for one example, is also found in [2].

When derivatives are available, one might ask why a direct search method would be chosen over the faster gradient-based or Newton-based methods. The primary reason is that gradient-based and Newton-based methods tend to converge quickly to local minima. The user may be willing to sacrifice the cost of additional function evaluations that direct search methods require in the hope of obtaining a better local minimum. Also, as will be shown, the algorithm shown here can make use of derivatives when some, but not all, partial derivatives are known, or when certain directional derivatives are known. This is of great benefit in some engineering applications where computing some directional derivatives is expensive and computing others are not. In this case, use of the inexpensive derivatives can speed convergence without sacrificing much in computational cost.

For the sake of simplicity, the ideas in this chapter are applied to the basic GPS algorithm, discussed in Section 4.1, that is used to solve the optimization problem given in (4.1). The ideas are easily extendable to bound and linear constrained MVP problems; however, applying them to NLP or MVP problems with nonlinear

constraints is more difficult, and remains an open area of research. The current implementation used in the NOMADm software (see Chapter 7) is a variant of the feasible directions method in which directions are constructed so that the resulting trial points remain on the GPS mesh.

8.2 The Basic GPS Algorithm With Derivative Information

The main idea of this chapter is that whenever the gradient of the objective function $\nabla f(x_k)$ at the current iterate x_k is available, it can be used to *prune* (or eliminate from consideration) the ascent directions from the poll set. The pruned set of polling directions, denoted $D_k^p \subseteq D_k$, is defined as follows.

Definition 8.1 Given a positive spanning set D_k, the *pruned set of polling directions*, denoted by $D_k^p \subset D_k$, is given by

$$D_k^p \;=\; \{d \in D_k : d^T \nabla f(x_k) \leq 0\}$$

when the gradient $\nabla f(x_k)$ is known, or by

$$D_k^p \;=\; D_k \setminus \{d \in V_k : f'(x_k; d) > 0\},$$

when incomplete derivative information is known, where $V_k \subset D_k$ is the set of directions in which the directional derivative is known. In the case where the gradient is known, $-\nabla f(x_k)$ is said to *prune* D_k to D_k^p.

Note that the pruning operation depends only on x_k, D_k, and the availability of the gradient, and is independent of Δ_k. We can now define the *pruned poll set* at iteration k as

$$P_k^p \;=\; \{x_k + \Delta_k d : d \in D_k^p\}. \tag{8.1}$$

The POLL step of the algorithm evaluates the constraints and objective at points in P_k^p until an improvement is found. If an improvement is not found, then the incumbent solution x_k is said to be a *mesh local optimizer* with respect to the pruned poll set. Notice that if all the points in the pruned poll set lie outside X, then f would not be evaluated at all in order to conclude that x_k is a mesh local optimizer. We have seen large savings from this fact in preliminary tests.

The GPS algorithm that uses derivative information is presented in Figure 8.1.

Generalized Pattern Search with Derivative Information

Initialization: Let x_0 be such that $f(x_0)$ is finite, and let M_0 be the mesh defined by $\Delta_0 > 0$ and x_0 (see (4.2)).

For $k = 1, 2, \ldots$, perform the following:

1. SEARCH **step:** Employ some finite strategy seeking an improved mesh point; *i.e.*, $x_{k+1} \in M_k$ such that $f(x_{k+1}) < f(x_k)$.

2. POLL **step:** If the SEARCH step fails to find an improved mesh point, evaluate f at points in the pruned poll set P_k^p until an improved mesh point x_{k+1} is found (or until done).

3. **Update:** If SEARCH or POLL finds an improved mesh point, set Δ_{k+1} according to (4.5); otherwise, set $x_{k+1} = x_k$ and set Δ_{k+1}, according to (4.6).

Figure 8.1   GPS Algorithm with Derivative Information

Clearly, if the gradient is available and is zero, then the algorithm can be stopped with a solution satisfying first order optimality conditions (a similar stopping criterion could be devised when the norm of the gradient is small enough).

If the gradient is available and is nonzero, Theorem 3.3 (also see [38]) guarantees that the negative gradient must prune at least one column of D_k, since there must be at least one column for which $d^T \nabla f(x_k) > 0$. Moreover, (by the same theorem) it

could prune as many as $|D_k| - 1$ directions (*i.e.*, all but one), leading to considerable savings for an expensive function. In section 8.3, we show how to systematically construct such a set by a special choice of D and D_k.

Of course, we expect that our pruning operation will yield more iterates with POLL steps that fail to find an improved mesh point, but this should happen less frequently as the mesh size parameter gets smaller. Furthermore, the role of the POLL step is mainly to ensure that the current iterate is a mesh local optimizer. This means that the string of successful searches on the current mesh is halted and searching can start afresh on a finer mesh. But for a sufficiently small mesh parameter, if our pruned POLL fails, then the "unpruned" POLL would also fail. This suggests that derivatives might be more useful when the mesh parameter is small and the POLL step would not be expected to get out of the current basin. Of course, the SEARCH can always move us into a deeper basin.

8.3   Using Derivative Information to Prune Well

Each iteration of a GPS algorithm is dependent upon the choice of a positive spanning set D_k, selected from the columns of the larger finite positive spanning set D. Here it is useful to think of D as preselected, but in fact, it can be modified finitely many times at the discretion of the user.

In this section, we first consider a simple measure of the richness of the set D. Theorem 3.3 shows that D is a positive spanning set for $\Re^n$ if and only if, for every $v \in \Re^n$, there exists $d \in D$ such that $v^T d > 0$. This says that a positive spanning set has at least one member in every half space. We are interested in D so rich in directions that, for every half space, we can select a subset of vectors in D that positively spans $\Re^n$, only one of which lies in that half space. We will show that no positive basis D can have it, and we will prove that $\mathbb{D} = \{-1, 0, 1\}^n$ has this

property. We build up to this point by considering some useful finer measures of directional richness in D. An important observation is that when gradients are not available, positive bases are useful in GPS because they give small poll sets [93], but when derivative information is available, positive bases may be a poor choice because a richer choice of directions makes it easier to isolate a descent direction and prune to a smaller poll set.

The following definition is consistent with the earlier Definition 8.1, although it is more general in keeping with the applications to follow.

Definition 8.2 (i) Given a positive spanning set $D \subset \Re^n$ and a nonzero vector $v \in \Re^n$, let $D(v) \subset D$ be a positive spanning set that minimizes the cardinality of the pruned set $D^p(v) = \{d \in D(v) : d^T v \geq 0\}$ (that cardinality is denoted $\rho(D, v)$). Then the vector v is said to *prune D to* $\rho(D, v)$ vectors.

(ii) Let $\rho(D) = \max_{v \in \Re^n \setminus \{0\}} \rho(D, v)$. Then the positive spanning set D is said to *prune to* $\rho(D)$ vectors.

First we collect some results on $\rho(D, v)$. From the discussion following Definition 8.1, the reader will see that $\rho(D, -\nabla f(x_k))$ is the minimal number of polling directions that can be found by pruning choices of $D_k = D(-\nabla f(x_k)) \subset D$. Consequently, the cardinality of the pruned poll set P_k^p constructed from $D_k^p = D^p(-\nabla f(x_k)) \subset D_k \subset D$ is minimal. We will say that P_k^p is a *minimal poll set*.

These results imply that the richer the directional choice in a positive spanning set D, the more sagacity can be employed to ensure that D_k can be extensively pruned. Of course, if D is not only a positive spanning set, but also a positive basis, then one would expect to allow less pruning since $D(v) = D$ for every choice of v. Indeed, it is not surprising for a positive basis D that $\rho(D)$ grows with the dimension n.

Proposition 8.3 If D is a positive spanning set, then $1 \leq \rho(D, v) \leq \rho(D) \leq |D| - 1$ for all nonzero $v \in \Re^n$. Moreover, if D is a positive basis, then $\frac{n+1}{2} \leq \frac{|D|}{2} \leq \rho(D) < |D| \leq 2n$.

Proof. The first assertion is direct, because for any $v \neq 0$ and any choice of positive spanning set, Theorem 3.3 guarantees that there must be at least one member of the set that makes a positive inner product with v, and another that makes a negative one. Thus, every possible $D^p(v)$ contains at least one element and at most $|D| - 1$.

This argument also guarantees that $\rho(D) < |D|$. To derive a lower bound on $\rho(D)$, we simply consider v and $-v$. Since no proper subset of a positive basis is a positive spanning set, the only possible choice for $D(v)$ and for $D(-v)$ is D. Thus, $D^p(v) \cup D^p(-v) = D$, and so $2\rho(D) \geq \rho(D, v) + \rho(D, -v) \geq |D|$. The other inequalities follow from [38], where it is shown that $2n \geq |D| \geq n+1$ for any positive basis. ∎

Finally, we use the notation and discussion above to state the following.

Proposition 8.4 If $\nabla f(x_k)$ is available at iteration k of the GPS algorithm, then the poll set at x_k has minimal cardinality $\rho(D, -\nabla f(x_k))$, which is an upper bound on the number of function evaluations required to execute the POLL step using a minimal poll set.

Proof. The proof follows directly from Definition 8.2 and from the discussion following it. ∎

In the next section, we propose a set of directions rich enough so that when the gradient is available the upper bound on the number of function evaluations in the POLL step (see Proposition 8.4) is one, thus P_k^p contains a single element.

8.3.1 Pruning with the Gradient

We show here that the set $\mathbb{D} = \{-1, 0, 1\}^n$ prunes to a singleton, *i.e.*, for any non-zero vector $v \in \Re^n$, there exists a positive spanning set $\mathbb{D}(v) \subset \mathbb{D}$ such that the subset of vectors of $\mathbb{D}(v)$ that makes a nonnegative inner product with v consists of a single element. The key lies in the construction of $\mathbb{D}(v)$, which is done by taking the union of the ascent directions of $\mathbb{D}$ with an element $\hat{d}$ of $\mathbb{D}$ that satisfies the properties,

$$\text{if } v_i = 0, \quad \text{then} \quad \hat{d}_i = 0, \tag{8.2}$$

$$\text{if } v_i \neq 0, \quad \text{then} \quad \hat{d}_i \neq -\text{sign}(v_i), \tag{8.3}$$

$$\text{if } |v_i| = \|v\|_\infty, \quad \text{then} \quad \hat{d}_i = \text{sign}(v_i), \tag{8.4}$$

for every $i \in N$, with the convention that $\text{sign}(0) = 0$. Our construction will be such that $\hat{d}$ will be the *only* unpruned member of $\mathbb{D}(v)$. The following technical lemma is necessary for the proof of Proposition 8.6.

Lemma 8.5 If $\delta > 1$ then

$$\frac{\delta - \sqrt{\delta^2 - 1}}{\sqrt{\delta^2 - 1}} > \frac{\sqrt{\delta^2 + 1} - \delta}{2\delta - \sqrt{\delta^2 + 1}}.$$

Proof. If $\delta > 1$, then the above denominators are non-negative and the following inequalities are equivalent to the one above:

$$
\begin{aligned}
(\delta - \sqrt{\delta^2 - 1})(2\delta - \sqrt{\delta^2 + 1}) \ &> \ (\sqrt{\delta^2 + 1} - \delta)(\sqrt{\delta^2 - 1}) \\
2\delta^2 - \delta\sqrt{\delta^2 + 1} \ &> \ \sqrt{\delta^2 - 1}\left(\sqrt{\delta^2 + 1} - \delta + 2\delta - \sqrt{\delta^2 + 1}\right) \\
2\delta \ &> \ \sqrt{\delta^2 + 1} + \sqrt{\delta^2 - 1} \\
2\delta^2 \ &> \ 2\sqrt{\delta^4 - 1}.
\end{aligned}
$$

This last inequality obviously holds for any $\delta > 1$. ∎

We first show the existence of vectors in $\mathbb{D}$ satisfying properties (8.2)–(8.4). This means that there are various choices for $\mathbb{D}(v)$. Three different choices for the single unpruned direction are given, and we will discuss their significance below.

Proposition 8.6 Let $\mathbb{D} = \{-1, 0, 1\}^n$. For any nonzero $v \in \Re^n$, properties (8.2)–(8.4) are satisfied by each vector $d^{(1)}, d^{(2)}$, and $d^{(\infty)}$, where for each $i \in N$,

$$d_i^{(1)} = \begin{cases} \operatorname{sign}(v_i) & \text{if } |v_i| = \|v\|_\infty, \\ 0 & \text{otherwise}, \end{cases} \tag{8.5}$$

$$d^{(2)} \in \arg\max_{d \in \mathbb{D} \setminus \{0\}} \frac{v^T d}{\|d\|_2}, \tag{8.6}$$

$$d_i^{(\infty)} = \operatorname{sign}(v_i). \tag{8.7}$$

Proof. Let v be a nonzero vector in $\Re^n$. The proof is trivial for $d^{(1)}$ and for $d^{(\infty)}$. The remainder of the proof is for $d^{(2)}$. Observe that since $d^{(\infty)} \in \mathbb{D} \setminus \{0\}$, the optimal value of the maximization operator in (8.6) is strictly positive.

To verify (8.3), suppose that $d_i^{(2)} = -\operatorname{sign}(v_i)$ for some index $i \in N$ with $v_i \neq 0$. Define $\hat{d}_\ell = d_\ell^{(2)}$ for $\ell \in N \setminus \{i\}$ and $\hat{d}_i = \operatorname{sign}(v_i)$. The optimality of $d^{(2)}$ is contradicted:

$$\frac{v^T d^{(2)}}{\|d^{(2)}\|_2} = \frac{1}{\|d^{(2)}\|_2} \sum_{\ell \in N \setminus \{i\}} v_\ell d_\ell^{(2)} - |v_i| < \frac{1}{\|d^{(2)}\|_2} \sum_{\ell \in N \setminus \{i\}} v_\ell d_\ell^{(2)} + |v_i| = \frac{v^T \hat{d}}{\|\hat{d}\|_2}.$$

To verify (8.2), suppose that $v_i = 0 \neq d_i^{(2)}$ for some index $i \in N$. Define $\hat{d}_\ell = d_\ell$ for $\ell \in N \setminus \{i\}$ and $\hat{d}_i = 0$. Note that (8.3) guarantees that $\hat{d} \in \mathbb{D} \setminus \{0\}$, thus $\|\hat{d}\|_2^2 = \|d^{(2)}\|_2^2 - 1 > 0$. The optimality of $d^{(2)}$ is contradicted:

$$0 < \frac{v^T d^{(2)}}{\|d^{(2)}\|_2} = \frac{1}{\|d^{(2)}\|_2} \sum_{\ell \in N \setminus \{i\}} v_\ell d_\ell^{(2)} < \frac{1}{\sqrt{\|d^{(2)}\|_2^2 - 1}} \sum_{\ell \in N \setminus \{i\}} v_\ell d_\ell^{(2)} = \frac{v^T \hat{d}}{\|\hat{d}\|_2}.$$

To verify (8.4), we must first show that two other properties hold; namely, for any $i, j \in N$,

(a) if $|v_i| > |v_j|$, then $|d_i^{(2)}| \geq |d_j^{(2)}|$.

(b) If $|v_i| = |v_j|$, then $|d_i^{(2)}| = |d_j^{(2)}|$.

To prove (a), let $|v_i| > |v_j|$ for some $i, j \in N$. Suppose that $|d_i^{(2)}| < |d_j^{(2)}|$; i.e., $d_i^{(2)} = 0, d_j^{(2)} = \text{sign}(v_j) \neq 0$. Define $\hat{d}_\ell = d_\ell^{(2)}$ for $\ell \in N \setminus \{i, j\}$ and $\hat{d}_i = \text{sign}(v_i)$ and $\hat{d}_j = 0$. Then we have $\|\hat{d}\|_2 = \|d^{(2)}\|_2$ and

$$v^T \hat{d} \;=\; \sum_{\ell \in N \setminus \{i,j\}} v_\ell d_\ell^{(2)} + |v_i| \;>\; \sum_{\ell \in N \setminus \{i,j\}} v_\ell d_\ell^{(2)} + |v_j| \;=\; v^T d^{(2)},$$

which contradicts the definition of $d^{(2)}$.

To prove (b), let $|v_i| = |v_j|$ for some $i, j \in N$. If $v_i = v_j = 0$, then $d_i^{(2)} = d_j^{(2)} = 0$. Now consider the case where both v_i and v_j are nonzero, and suppose that $|d_i^{(2)}| \neq |d_j^{(2)}|$; without any loss of generality, suppose that $d_i^{(2)} = 0, d_j^{(2)} = \text{sign}(v_j) \neq 0$. Let $\delta = \|d^{(2)}\|_2$. Define $\hat{d}_\ell = d_\ell^{(2)}$ for $\ell \in N \setminus \{i\}$ and $\hat{d}_i = \text{sign}(v_i)$. If $\delta = 1$, then a contradiction follows:

$$\frac{v^T d^{(2)}}{\|d^{(2)}\|_2} \;=\; |v_j| \;<\; \sqrt{2}|v_j| = \frac{|v_i| + |v_j|}{\sqrt{2}} \;=\; \frac{v^T \hat{d}}{\|\hat{d}\|_2}.$$

If $\delta > 1$, then set $\sigma = \sum_{\ell \in N \setminus \{i,j\}} v_\ell d_\ell^{(2)} > 0$. Since $d^{(2)}$ is an optimal solution of (8.6), then

$$\frac{\sigma}{\sqrt{\delta^2 - 1}} \;\leq\; \frac{v^T d^{(2)}}{\|d^{(2)}\|_2} = \frac{\sigma + |v_j|}{\delta} \quad \Longleftrightarrow \quad \frac{|v_j|}{\sigma} \;\geq\; \frac{\delta - \sqrt{\delta^2 - 1}}{\sqrt{\delta^2 - 1}}.$$

The previous lemma leads to a contradiction:

$$\frac{|v_j|}{\sigma} \;>\; \frac{\sqrt{\delta^2 + 1} - \delta}{2\delta - \sqrt{\delta^2 + 1}} \quad \Longleftrightarrow \quad \sigma(\sqrt{\delta^2 + 1} - \delta) \;<\; |v_j|(2\delta - \sqrt{\delta^2 + 1})$$

$$\Longleftrightarrow \quad \frac{v^T d^{(2)}}{\|d^{(2)}\|_2} = \frac{\sigma + |v_j|}{\delta} \;<\; \frac{\sigma + |v_i| + |v_j|}{\sqrt{\delta^2 + 1}} = \frac{v^T \hat{d}}{\|\hat{d}\|_2}.$$

Now, with (a) and (b) proven, let $i \in N$ satisfy $|v_i| = \|v\|_\infty$. By (a) and (b), we have $|d_i^{(2)}| \geq |d_j^{(2)}|$ for all $j \in N$. Since $d^{(2)} \neq 0$, it follows that $d_j^{(2)} = \text{sign}(v_j)$. ∎

Since the vectors $d^{(2)}$ and $d^{(\infty)}$ can be used to prune $\mathbb{D}$ to a singleton, and since $d^{(2)}$ is the vector in $\mathbb{D}$ that makes the smallest angle with v, it is tempting to conjecture that any vector in $\mathbb{D}$ making a smaller angle with v than $d^{(\infty)}$ can be use to prune to a singleton. However, this is not the case, as the following counterexample shows.

Example 8.7 In $\Re^5$, let $v = (1.1, 1, .01, .01, .01)$. Then $d^{(\infty)}$ is the vector of ones in $\Re^5$, and $\frac{v^T d^{(\infty)}}{\|d^{(\infty)}\|} = \frac{2.15}{\sqrt{5}} < 1$. Observe that $\mathbb{A}(v)$ contains no vectors with a 1 in the first component; otherwise, its inner product with v must be positive.

Now let $\hat{d} = e_2$. Then $\frac{v^T \hat{d}}{\|\hat{d}\|} = \frac{1}{\sqrt{1}} = 1 > \frac{v^T d^{(\infty)}}{\|d^{(\infty)}\|}$. Thus, $\hat{d} \neq d^{(2)}$ makes a smaller angle with v than $d^{(\infty)}$ does, but $\{\hat{d}\} \cup \mathbb{A}(v)$ does not positively span $\Re^5$ because it cannot positively generate e_1. ■

The next theorem shows that for any nonzero vector $v \in \Re^n$, any direction $\hat{d}$ satisfying properties (8.2)–(8.4) can be completed into a positive spanning set by adding the directions in

$$\mathbb{A}(v) = \left\{ d \in \mathbb{D} : v^T d < 0 \right\}, \tag{8.8}$$

and consequently, v prunes $\mathbb{D}$ to the single vector $\{\hat{d}\}$.

Theorem 8.8 The set $\mathbb{D} = \{-1, 0, 1\}^n$ has $\rho(\mathbb{D}) = 1$.

Proof. Let $v \in \Re^n$ be nonzero, and let $\hat{d} \in \mathbb{D}$ be any vector satisfying the properties (8.2)–(8.4), and define $\mathbb{D}(v)$ to be the union of $\{\hat{d}\}$ with the set $\mathbb{A}(v)$ of directions in $\mathbb{D}$ that make negative inner products with $\hat{d}$.

Theorem 3.3 states that $\mathbb{D}(v) = \{\hat{d}\} \cup \mathbb{A}(v) \subset \mathbb{D}$ positively spans $\Re^n$ if and only if, for any nonzero $b \in \Re^n$ there exists a vector d in $\mathbb{D}(v)$ such that $b^T d > 0$. Let $b \in \Re^n$ be a nonzero vector. If $b^T \hat{d} \neq 0$, then clearly either of $\pm \hat{d}$ makes a positive inner

product with b, in which case, we are done since both are in $\mathbb{D}(v)$. Thus, consider the situation where $b^T \hat{d} = 0$. The analysis is divided in two disjoint cases.

Case 1: $b_i v_i < 0$ for some index $i \in N$. Set $d_j = 0$ for $j \in N \setminus \{i\}$ and $d_i = \text{sign}(b_i)$. It follows that d belongs to $\mathbb{A}(v) \subset \mathbb{D}(v)$ and makes a positive inner product with b since

$$v^T d = v_j d_j = v_j \text{sign}(b_j) = -v_j \text{sign}(v_j) = -|v_j| < 0, \quad \text{and}$$
$$b^T d = b_j d_j = b_j \text{sign}(b_j) = |b_j| > 0.$$

Case 2: $b_i v_i \geq 0$ for all $i \in N$. From (8.2) and (8.3), we have $b_i \hat{d}_i \geq 0$ for all $i \in N$, and since $b^T \hat{d} = 0$, it follows that $b_i \hat{d}_i = 0$ for all $i \in N$. Let $i, j \in N$ be such that $|v_i| \geq |v_\ell|$ for all $\ell \in N$ and $b_j \neq 0$. Then $b_i = 0$, $\hat{d}_j = 0$, and by (8.4), $\hat{d}_i = \text{sign}(v_i) \neq 0$. Furthermore, since $\hat{d}_j = 0$, $|v_j| < |v_i|$. Set $d_\ell = 0$ for $\ell \in N \setminus \{i, j\}$ and $d_i = -\text{sign}(v_i)$ and $d_j = \text{sign}(v_j)$. It follows that d belongs to $\mathbb{A}(v) \subset \mathbb{D}(v)$ and makes a positive inner product with b since

$$v^T d = v_i d_i + v_j d_j = -v_i \text{sign}(v_j) + v_j \text{sign}(b_j) \leq -|v_i| + |v_j| < 0, \quad \text{and}$$
$$b^T d = b_i d_i + b_j d_j = b_j \text{sign}(b_j) = |b_j| > 0.$$

Thus $\mathbb{D}^p(v) = \{\hat{d}\}$, and the proof is complete since $v \neq 0$ was arbitrary. ∎

We now apply these results to the determination of the poll set in the GPS algorithm. Of course, for $v = -\nabla f(x_k)$, the set $\mathbb{A}(-\nabla f(x_k))$ would contain all ascent directions, which $-\nabla f(x_k)$ then prunes away. Therefore, the ascent directions in $\mathbb{A}(-\nabla f(x_k))$ do not need to be explicitly constructed.

The notation established earlier becomes clear. For $v = -\nabla f(x_k)$, the vectors $d^{(1)}$ and $d^{(\infty)}$ are the negatives of the normalized ℓ_1 and ℓ_∞ gradients of f at x_k, respectively (see [28]), while $d^{(2)}$ is the vector in $\mathbb{D}$ that makes the smallest angle with $-\nabla f(x_k)$.

Theorem 8.9 If $\nabla f(x_k)$ is available at iteration k of the GPS algorithm with positive spanning set $D = \mathbb{D}$ and if $D_k = \mathbb{D}(-\nabla f(x_k)) = \{\hat{d}\} \cup \mathbb{A}(-\nabla f(x_k))$ with $\hat{d}$ satisfying the properties (8.2)–(8.4), then a single function evaluation is required for the POLL step.

Proof. Theorem 8.8 and Proposition 8.4 with $v = -\nabla f(x_k)$ guarantee the result. ∎

Thus, choosing the rich set of directions $\mathbb{D}$ and using the gradient information allows us to evaluate the barrier objective function f_X at a single poll point $x_k + \Delta_k \hat{d}$. This may not require *any* evaluations of the objective function f if the trial point lies outside of X, or obviously if $\nabla f(x_k) = 0$.

8.3.2 Pruning with an Approximation of the Gradient

In many engineering applications, derivatives are approximated or inaccurately computed without much additional cost during the computation of the objective. Let us define a measure of the quality of an approximation of a vector.

Definition 8.10 Let g be a nonzero vector in $\Re^n$ and $\epsilon \geq 0$. Define $J^\epsilon(g) = \{i \in N : |g_i| + \epsilon \geq \|g\|_\infty\}$, and for every $i \in N$ set

$$d_i^\epsilon(g) = \begin{cases} \text{sign}(g_i) & \text{if } i \in J^\epsilon(g) \\ 0 & \text{otherwise.} \end{cases}$$

The vector g is said to be an $\epsilon-$approximation to the large components of a non-zero vector $v \in \Re^n$ if $i \in J^\epsilon(g)$ whenever $|v_i| = \|v\|_\infty$ and if $\text{sign}(g_i) = \text{sign}(v_i)$ for every $i \in J^\epsilon(g)$.

Note that if $\epsilon = 0$, then $d^\epsilon(g)$ is identical to $d^{(1)}$ in (8.5), and if $\epsilon = \|g\|_\infty$, then then $d^\epsilon(g)$ is identical to $d^{(\infty)}$ in (8.7). The following result implicitly provides a sufficiency condition on the quality of the approximation.

Proposition 8.11 Let $\epsilon \geq 0$ and g be an $\epsilon-$approximation to the large components of a non-zero vector $v \in \Re^n$. Then $d^\epsilon(g)$ satisfies properties (8.2)–(8.4).

Proof. Let $\epsilon \geq 0$ and $g \neq 0$ be an $\epsilon-$approximation to the large components of $v \neq 0 \in \Re^n$. The set $J^\epsilon(g)$ is nonempty and contains every i for which $|v_i| = \|v\|_\infty$. If $i \in J^\epsilon(g)$ then $d_i^\epsilon(g) = \text{sign}(g_i) = \text{sign}(v_i)$, and therefore properties (8.2)–(8.4) are satisfied. If $i \notin J^\epsilon(g)$ then the properties are trivially satisfied. ∎

The next result establishes a mild accuracy requirement for an approximation of the negative gradient to prune $\mathbb{D}(-\nabla f(x_k))$ to a singleton.

Lemma 8.12 Let $\epsilon \geq 0$ and g_k be an $\epsilon-$approximation to the large components of $\nabla f(x_k) \neq 0$. If there exists a vector $\hat{d}$ satisfying (8.2)–(8.4) for both $v = -\nabla f(x_k)$ and $v = -g_k$, then the set $\{\hat{d}\} \bigcup \mathbb{A}(-\nabla f(x_k))$ positively spans $\Re^n$ and prunes to $\{\hat{d}\}$.

Proof. This follows directly from the proof of Theorem 8.8. ∎

The significance of this result resides in the wide latitude allowed for the approximation. For example, if $d^{(\infty)}$ with $v = -g_k$ is used (see (8.7)) to obtain $\hat{d}$, then we only need the components of the approximation g_k to match signs with those of the true gradient $\nabla f(x_k)$ in order to prune to a singleton. If $d^{(1)}$ with $v = -g_k$ is used (see (8.5)) to obtain $\hat{d}$, then we only need the component of largest magnitude of g_k to have the same sign as that of $\nabla f(x_k)$.

The following result shows that if an approximation to the gradient matches signs with the true gradient for all components of sufficiently large magnitude, then the approximation prunes the set of poll directions to a singleton.

Theorem 8.13 Let $\epsilon \geq 0$. At iteration k of the GPS algorithm with $D = \mathbb{D} = \{-1, 0, 1\}^n$, let $g_k \neq 0$ be an $\epsilon-$approximation of $\nabla f(x_k) \neq 0$.

Then the set of directions $D_k = \{d^\epsilon(g)\} \bigcup \mathbb{A}(-\nabla f(x_k))$ positively spans $\Re^n$. Thus, $D_k^p = \{d^\epsilon(g)\}$.

Proof. This follows directly by combining the fact that $d^\epsilon(g)$ is the same if constructed from $v = \nabla f(x_k)$ or $v = g_k$ and by Lemma 8.12. ∎

Observe that when $\epsilon = 0$ or $\epsilon = \|g\|_\infty$, then Theorem 8.13 reduces to Theorem 8.9 with $d^\epsilon(g) = d^{(1)}$ or $d^\epsilon(g) = d^{(\infty)}$, respectively.

8.3.3 Pruning with Incomplete Derivative Information

It is possible that in some instances and some iterations, the entire gradient is not available, but a few partial or directional derivatives might be known. This situation may occur for example when $f(x, y) = g(x, y) \times h(y)$ where $g(x, y)$ is analytically given, but the structure of $h(y)$ is unknown. In such a case the partial derivatives of f can be computed with respect to x but not with respect to y.

If at iteration k, the directional derivative $f'(x_k; d)$ exists, is available, and is nonnegative, then polling in the direction d from x_k will fail if the mesh size parameter is small enough. This leads to the following result.

Proposition 8.14 Let $D_k \subseteq D$ be a positive spanning set and V_k be the subset of directions d in D_k for which the directional derivative $f'(x_k; d) > 0$ is known. Then the pruned set of directions is $D_k^p = D_k \backslash V_k$ and contains $|D_k| - |V_k|$ directions.

Proof. The result follows from the fact that the directional derivative $f'(x_k; d)$ equals $d^T \nabla f(x_k)$. ∎

Note that when the gradient exists, if $f'(x_k; d)$ is nonpositive, then $f'(x_k; -d)$ is nonnegative. The most typical application of incomplete gradient information is

when some, but not all, partial derivatives are known. The spanning set $\mathbb{D}$ can be used to reduce as much as possible the size of the pruned poll set. The approach for doing so can be outlined as follows:

Pruning Operation for Incomplete Gradients

For $x_k \in \Re^n$ and a set of nonzero partial derivatives $\frac{\partial f}{\partial x_{i_j}}(x_k)$, $j = 1, 2, \ldots, t \leq n$,

1. Set $V_k = \{s_{i_j} e_{i_j} : j = 1, 2, \ldots, t\} \subset \mathbb{D}$, where $s_{i_j} = \text{sign}(\frac{\partial f}{\partial x_{i_j}}(x_k))$, $j = 1, 2, \ldots, t$,

2. Set $W_k = \{e_\ell : \ell \in L\}$, where $L = N \setminus \{i_1, i_2, \ldots, i_t\}$,

3. Set $u = -\sum_{j=1}^{t} s_{i_j} e_{i_j} - \sum_{\ell \in L} e_\ell$,

4. Set $D_k = V_k \cup W_k \cup \{u\}$, and prune D_k to $D_k^p = W_k \cup \{u\}$

For this construction, we can prove the following result:

Corollary 8.15 Let the partial derivatives at iteration k, $\frac{\partial f}{\partial x_{i_j}}(x_k)$ for $j = 1, 2, \ldots, t$ be available and all nonzero. If $\nabla f(x_k)$ exists, then the pruning operation yields a pruned set D_k^p with $n + 1 - t$ directions.

Proof. By the construction above, it is clear that $B = V_k \cup W_k$ forms a basis for $\Re^n$, and $u = -\sum_{i=1}^{n} b_i$, where $b_i \in B, i = 1, 2, \ldots, n$. Thus $D_k = B \cup \{u\}$ forms a positive basis for $\Re^n$ with $|D_k| = n + 1$. Furthermore, since $\nabla f(x_k)$ exists, for any $v_j \in V_k$, we have $f'(x_k; v_j) = \nabla f(x_k)^T v_j = \nabla f(x_k)^T s_{i_j} e_{i_j} = |\frac{\partial f}{\partial x_{i_j}}(x_k)| > 0$. Since V_k has t directions, the result follows from Proposition 8.14. ∎

The following example illustrates this result.

Example 8.16 Consider $f : \Re^3 \to \Re$ with $\frac{\partial f}{\partial x_1}(x_k) > 0$ and $\frac{\partial f}{\partial x_2}(x_k) < 0$, where f is continuously differentiable at x_k. Then, with $\mathbb{D} = \{-1, 0, 1\}^n$,

by defining D_k as

$$D_k = \begin{bmatrix} 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix},$$

the pruned set D_k^p can be constructed using only the two last columns of D_k.

The results derived in this section are compatible with the previous ones. Indeed, if the full gradient is available and if all its component are non-zero, then using $\mathbb{D}$ the pruned set requires $n + 1 - n = 1$ direction, $i.e.$, $\rho(\mathbb{D}) = 1$.

8.3.4 Pruning with Linear Constraints

A similar strategy can be used in the bound or linearly constrained case. Recall that $X = \{x \in \Re^n : Ax \leq b\}$ defines the feasible region, where $A \in \mathcal{Q}^{m \times n}$ and $b \in \Re^m$. Set $M = \{1, 2, \ldots, m\}$ and let a_j^T be the j-th row of A for $j \in M$.

For any $\epsilon > 0$ and $x \in X$, define $\mathcal{A}_\epsilon(x) = \{j \in M : a_j^T x \geq b_j - \epsilon\}$, the set of ϵ-active constraints (as used in [95]). The set of positive spanning directions D is implicitly defined to contain all tangent cone generators of all points on all faces of the polytope X. Obviously, this set is never explicitly constructed. However, for any $x \in X$ and for a given $\epsilon > 0$, we will need to construct, as suggested in [95], the set $T(x) \subset D$ of tangent cone generators to all the ϵ-active constraints. With this construction, if $D_k \subset D$ is a positive spanning set containing $T(x_k)$, then D_k will conform to the boundary of X for $\epsilon > 0$.

If the gradient is known, then we can find a bound for the minimal cardinality of the pruned set through the following approach and subsequent proposition.

Pruning Operation under Linear Constraints

For $x_k \in \Re^n$ with known gradient $\nabla f(x_k)$,

1. Let $T(x_k)$ be the set of tangent cone generators to all ϵ-active constraints.

2. Set $D' = T(x_k) \cup \{d \in \mathbb{D} : \nabla f(x_k)^T d > 0\}$.

3. If D' positively spans $\Re^n$, set $D_k = D'$; otherwise, set $D_k = D' \cup \{\hat{d}\}$, where $\hat{d}$ satisfies properties (8.2)–(8.4).

4. Prune D_k to D_k^p.

Proposition 8.17   Let $\epsilon > 0$. If $\nabla f(x_k)$ is known at iteration k, then the pruning operation yields a pruned set D_k^p containing at most $|T(x_k)| + 1$ directions.

Proof. By construction, in the worst case, $D_k = T(x_k) \cup \{\hat{d}\} \cup \{d \in \mathbb{D} : \nabla f(x_k)^T d > 0\}$, and $D_k^p = T(x_k) \cup \{\hat{d}\}$, whose cardinality is $|T(x_k)| + 1$. ∎

Again, this result is in agreement with previous ones: in the unconstrained case, or if there are no ϵ-active constraints, then $\mathcal{A}_\epsilon(x_k)$ is empty, reducing D_k^p to a singleton. Also, when $\{\hat{d}\}$ is used, then $\hat{d}$ points outside X for some boundary point within ϵ of x_k and if the trial point $x_k + \Delta_f \hat{d}$ does not belong to X, then f will not be evaluated at that trial point.

If only incomplete derivative information is available, the gradient cannot be used as strongly to prune the set of directions. Define $D' = T(x_k) \cup \{d \in V_k : f'(x_k; d) > 0\}$, then construct D_k by completing (if necessary) D' into a positive spanning set. As before, D_k^p is obtained by pruning ascent directions from D_k. By construction, all

directions in $\{d \in V_k : f'(x_k; d) > 0\}$ will get pruned, and some directions in $T(x_k)$ might get pruned.

8.4 Convergence Results

Except for a slight modification in one key theorem, convergence results for this method are identical to those of Audet and Dennis [6], and are detailed in [2]. Furthermore, they are summarized in Chapter 4, and proved in the more general MVP case in Chapter 5. Thus, convergence results are omitted here, except in the following theorem where a difference is noted.

But first, as in Chapter 4, we make the following assumptions:

A1: All iterates $\{x_k\}$ produced by the algorithm lie in a compact set.

A2: The set of directions $D = GZ$, as defined in (4.3), includes tangent cone generators for every point in X.

A3: The rule for selecting positive spanning sets D_k conforms to X for some $\epsilon > 0$.

Note that, as in Chapter 4, Assumption A2 is satisfied if if $G = I$ and the constraint coefficient matrix A is rational.

Theorem 8.18 Let $\hat{x}$ be the limit of a refining subsequence, and let d be any direction in D for which f at a POLL step was evaluated or pruned by the gradient for infinitely many iterates in the subsequence. Under assumptions A1-A3, if f is Lipschitz near $\hat{x}$, then the generalized directional derivative of f at $\hat{x}$ in the direction d is nonnegative, i.e., $f^\circ(\hat{x}; d) \geq 0$.

Proof. Let $\{x_k\}_{k \in K}$ be a refining subsequence with limit point $\hat{x}$ and $d \in D$ obtained as in the statement of the Theorem (finiteness of D ensures the existence of d). Since

f is locally Lipschitz near $\hat{x}$, we have from Clarke [31] by definition that:

$$f^\circ(\hat{x};d) \quad \equiv \quad \limsup_{y\to\hat{x},\ t\downarrow 0} \frac{f(y+td)-f(y)}{t} \quad \geq \quad \limsup_{k\in K} \frac{f(x_k+\Delta_k d)-f(x_k)}{\Delta_k} \quad . \quad (8.9)$$

First note that since f is Lipschitz near $\hat{x}$, it must be finite near $\hat{x}$. Since infeasible points are not evaluated by the algorithm, the hypothesis of f being evaluated or pruned infinitely many times in direction d guarantees that infinitely many right-hand quotients are defined.

The analysis is divided in two cases. First, consider the case where the gradient is evaluated only a finite number of times in the subsequence $\{x_k\}_{k\in K}$. Then these finite number of iterates may be ignored, and therefore, for k sufficiently large, all the POLL steps in the direction d, $x_k + \Delta_k d$, are feasible. If they had not been, then f would not have been evaluated. Thus, we have that infinitely many of the right hand quotients of (8.9) are defined. Then all of them must be nonnegative or else the corresponding POLL step would have found an improved mesh point, a contradiction (recall that refining subsequences are constructed from mesh local optimizers).

Second, consider the case where there is an infinite number of iterates in the subsequence where the gradient is used. Then there is a subsequence that converges to $\hat{x}$ for which $d^T\nabla f(x_k) \geq 0$ and thus the right hand side of (8.9) is bounded below by zero. ∎

8.5 Numerical Experiments

The algorithms described in this chapter were implemented in the Matlab® software, NOMADm [1], (discussed in Section 7.1), and applied to 18 problems from the CUTE [18] collection, as well as to a simple example to illustrate the value of a simple, but not trivial, SEARCH step.

Most of the CUTE problems we tested have multiple first-order stationary points, which is important if we are to compare the solutions found by various version of the algorithm. For each problem, performance of these algorithms is compared among several poll strategies. No SEARCH method is employed, and all problems are solved to within 10^{-4} accuracy or until 50,000 function evaluations are performed. (This was the reasonable limit of the NOMADm code.) A mathematical description of each problem is provided in Appendix A.

Specifically, Table 8.1 shows objective function value attained, number of function evaluations, and number of gradient evaluations for each problem. The seven poll strategies identified in the column headings of Table 8.1 differ in their choice of directions and are described as follows:

- **Stand$_{2n}$:** standard $2n$ directions, $D = [I, -I]$ with no gradient-pruning.

- **Stand$_{n+1}$:** standard $n + 1$ directions, $D = [I, -e]$ with no gradient-pruning, where e is the vector of ones.

- **Grad$_{2n}$:** gradient-pruned subset of the standard $2n$ directions.

- **Grad$_{n+1}$:** gradient-pruned subset of the standard $n + 1$ directions.

- **Grad$_{3^n}^{(1)}$:** gradient-pruned subset of $\mathbb{D} = \{-1, 0, 1\}^n$, pruned by $d^{(1)}$.

- **Grad$_{3^n}^{(2)}$:** gradient-pruned subset of $\mathbb{D} = \{-1, 0, 1\}^n$, pruned by $d^{(2)}$.

- **Grad$_{3^n}^{(\infty)}$:** gradient-pruned subset of $\mathbb{D} = \{-1, 0, 1\}^n$, pruned by $d^{(\infty)}$.

- **Grad$_{3^n, 2n}^{(2)}$:** a combination of Grad$_{3^n}^{(2)}$ and Grad$_{2n}$.

First, the example to illustrate the value of a very basic SEARCH step.

Example 8.19 We ran NOMADm on the problem,

$$\min \quad f(x, y) = x^3 + y^3 - 10(x^2 + y^2)$$

$$\text{s.t.} \quad -5 <= x <= 10$$

$$-5 <= y <= 10$$

with an initial point of $x_0 = (0.5, 0.5)$. The results were:

- **Stand**$_{2n}$ found $\hat{x} = (6.666, -5)$ with $f(\hat{x}) = -523$ in 94 evaluations of f and no evaluations of ∇f.

- **Grad**$_{3n}^{(\infty)}$ found $\hat{x} = (6.666, 6.666)$ with $f(\hat{x}) = -296$ in 33 evaluations of f and 17 evaluations of ∇f.

- **Grad**$_{3n}^{(\infty)}$ with only a two point feasible random search found $\hat{x} = (-5, -5)$ with $f(\hat{x}) = -750$ in 85 evaluations of f and 5 evaluations of ∇f.

Thus, **Stand**$_{2n}$ found a good local minimizer, and **Grad**$_{3n}^{(\infty)}$ without a search finds a higher local optimum. But, when we add the 2-point random search to **Grad**$_{3n}^{(\infty)}$, we find the global optimum.

For the CUTE problem results given in Table 8.1 below, numbers appearing in parentheses indicate that a second set of runs was performed for that problem, but with a slightly perturbed initial point. In these cases, if a particular entry contains no number in parentheses, then that value remained unchanged in the second run. The initial point was perturbed whenever the number of required function evaluations for a run was very small. This occurs when the choices of initial point and poll directions quickly drive the algorithm to an *exact* stationary point (*i.e.*, $\nabla f(x_k) = 0$). Since this phenomenon is not common in practice, we didn't want to make a particular strategy to appear gratuitously advantageous over another.

It should be noted that three problems, ALLINIT, ALLINITU, and EXPFIT do not have starting points associated with them. Since ALLINIT is the constrained version of ALLINITU, we chose the same reasonable and feasible initial point for both problems, while the vector of ones was chosen for EXPFIT. Also, the problem OSLBQP has an infeasible starting point with respect to one of its lower bounds. Our NOMADm software automatically restores any such variable to its nearest bound before proceeding with the algorithm. (In the case of general linear constraints, our software shifts an infeasible starting point to the nearest feasible point with respect to the Euclidean norm.)

In general, the gradient-pruned GPS methods converged in fewer function evaluations than the standard GPS methods, but they often terminated at a point with a higher function value (and vice versa), particularly if only one gradient-pruned descent direction was used in the POLL step. In fact, in no case did a standard method require fewer function evaluations than a "3^n" gradient method, and in only one case (MDHOLE, Stand_{n+1}) did a standard method require fewer function evaluations than its gradient-pruned counterpart with the same poll directions (e.g., Stand_{2n} versus Grad_{2n}, Stand_{n+1} versus Grad_{n+1}). Also, $\text{Grad}_{3^n,2n}^{(2)}$ was always at least as fast as Stand_{2n}.

However, the problems ALLINITU, MARATOSB, MEXHAT, and OSBORNEA are examples of cases where a gradient-pruned method converged to a lower point than a standard poll. In fact, $\text{Grad}_{3^n}^{(2)}$ (with only one function evaluation per iteration) achieved a lower function value for OSBORNEA than both standard poll methods despite fewer function evaluations.

The problem PALMER1 is an interesting example with typical behavior. The three highly-pruned Grad_{3^n} strategies converge the quickest, but to the much poorer solution of 43904 than the other methods. The methods that used the most directions,

Stand_{2n}, Grad_{2n}, and $\text{Grad}_{3^n,2n}^{(2)}$, converge to the best solution of 11760, with Grad_{2n} saving a considerable number of function evaluations over Stand_{2n}. The two $n + 1$ direction methods converge to a solution of 21166, with gradient-pruning again saving a significant number of function evaluations. Finally, the $\text{Grad}_{3^n,2n}^{(2)}$ strategy achieves the best optimal value faster than Stand_{2n}, but it wasn't able to match Grad_{2n} in function evaluations, perhaps because it requires slightly more function evaluations per POLL step.

Table 8.1 Numerical Results for Selected CUTE Test Problems.

ALLINIT, $n = 4$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	26.7643	26.7643	26.7643	26.7643	26.7643	26.7643	26.7643	26.7643
Function evaluations	85	85	47	47	47	56	47	56
Gradient evaluations	0	0	10	10	10	10	10	10

ALLINITU, $n = 4$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	6.9288	5.7444	6.9288	5.7661	5.8006	5.7812	9.2523	6.9293
Function evaluations	1628	425	846	133	30	65	25	650
Gradient evaluations	0	0	140	20	7	14	7	65

BARD, $n = 3$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	0.0082	0.0122	0.0082	0.0089	0.0089	0.0083	0.0091	0.0082
Function evaluations	11061	50000+	5963	50000+	2430	7799	1263	4871
Gradient evaluations	0	0	1640	16384	1018	2474	636	722

BOX2, $n = 3$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	0	0	0	0	0	0	0	0
Function evaluations	61	48	31	32	25	25	25	56
Gradient evaluations	0	0	2	2	8	8	8	4

BOX3, $n = 3$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	0	0	0	0	0.5656	0.2253	0.5656	0
Function evaluations	91	63	46	32	17	32	17	62
Gradient evaluations	0	0	2	2	2	2	2	2

DENSCHNA, $n = 2$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	0	0	0	0	0	0	0	0
Function evaluations	73(148)	47(156)	67(86)	47(92)	5(56)	2(23)	2(23)	2(83)
Gradient evaluations	0	0	4(22)	2(30)	3(20)	2(8)	2(8)	2(14)

DENSCHNB, $n = 2$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	0	0	0	0	0	0	0	0
Function evaluations	68(119)	130(95)	68(66)	87(53)	6(50)	4(28)	4(27)	4(71)
Gradient evaluations	0	0	3(15)	29(16)	3(16)	3(11)	3(10)	3(10)

DENSCHNC, $n = 2$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	0	0(0.0001)	0	0(0.001)	0	0	0	0
Function evaluations	75(119)	54(662)	67(68)	50(384)	7(87)	5(103)	4(62)	16(98)
Gradient evaluations	0	0	5(15)	4(168)	4(31)	3(33)	3(28)	5(16)

EXPFIT, $n = 2$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	0.2405	0.2406	0.2405	0.2406	0.2405	0.2405	0.2405	0.2405
Function evaluations	300	999	191	524	164	163	81	198
Gradient evaluations	0	0	66	251	66	69	37	47

MARATOSB, $n = 2$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	-1	0	-0.0041	0	-1	-1	-1	-1
Function evaluations	66	50	42	34	17	17	17	17
Gradient evaluations	0	0	3	3	2	2	2	2

MDHOLE, $n = 2$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	0	0	8130.9	0.0196	11127.6	8130.9	11127.6	8130.9
Function evaluations	37610	48	31	5043	15	31	15	46
Gradient evaluations	0	0	2	1516	1	2	1	2

MEXHAT, $n = 2$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	-0.0401	-0.0393	-0.0401	-0.0401	-0.0401	-0.0401	-0.0401	-0.0401
Function evaluations	350	69	203	31	32	35	20	285
Gradient evaluations	0	0	54	7	4	4	4	53

MEYER3, $n = 3$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	1692509	1446638	1692509	1446638	23591906	5469396	22772194	1692509
Function evaluations	2028	2581	1243	1439	395	1768	298	1703
Gradient evaluations	0	0	434	659	204	767	155	434

OSBORNEA, $n = 5$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	0.00106	0.00029	0.00106	0.00029	0.00185	0.00015	0.00089	0.00012
Function evaluations	2263	13300	1222	7783	496	1455	542	1770
Gradient evaluations	0	0	183	2042	286	528	314	237

OSBORNEB, $n = 11$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	0.04014	0.15192	0.04014	0.12229	0.04131	0.04088	0.0404	0.04014
Function evaluations	23191	50000+	11755	50000+	1606	3866	975	6574
Gradient evaluations	0	0	736	6479	777	1245	481	355

OSLBQP, $n = 8$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	6.25	6.25	6.25	6.25	6.25	6.25	6.25	6.25
Function evaluations	167	1355	115	472	5	5	5	8
Gradient evaluations	0	0	7	101	5	5	5	6

PALMER1, $n = 3$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	11760	21166	11760	21166	43904	43904	43904	11760
Function evaluations	1498	438	730	208	31	30	31	1013
Gradient evaluations	0	0	160	64	11	11	13	161

PALMER1A, $n = 4$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	116.6	7965.8	41.3	393.8	2262.7	4989.1	729.3	45.9
Function evaluations	50000+	50000+	50000+	50000+	50000+	50000+	50000+	50000+
Gradient evaluations	0	0	5636	9213	20128	17752	24999	4360

PALMER1B, $n = 2$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	4.0137	9010.0	3.4682	6.3366	9080.6	13474.1	5135.9	3.4671
Function evaluations	50000+	50000+	43899	50000+	50000+	50000+	50000+	44127
Gradient evaluations	0	0	7684	11450	20441	25001	25001	5149

PALMER1C, $n = 8$	Stand_{2n}	Stand_{n+1}	Grad_{2n}	Grad_{n+1}	$\text{Grad}_{3n}^{(1)}$	$\text{Grad}_{3n}^{(2)}$	$\text{Grad}_{3n}^{(\infty)}$	$\text{Grad}_{3n,2n}^{(2)}$
Final f-value	16948398	30078	12514913	30078	39201	39859	41284	799100
Function evaluations	50000+	36077	50000+	5483	2752	3788	252	50000+
Gradient evaluations	0	0	5535	2491	1033	1140	120	4230

Chapter 9

Conclusions and Recommendations

9.1 Summary and Conclusions

A new class of algorithms for solving mixed variable optimization problems with nonlinear constraints has been presented. It is a generalization of existing GPS algorithms in that, it reduces to that of [7] or [8] if there are, respectively, no nonlinear constraints or categorical variables. New convergence theorems have been proved, with results consistent with those of existing algorithms. Furthermore, the algorithm represents the first optimization algorithm with provable convergence properties that can be applied to MVP problems with general nonlinear constraints. This has been demonstrated in practice by the solution of the problem described in Chapter 7.

The following list describes the new contributions of this work:

- the concept of D-regularity and its relationship to the Clarke calculus given in Section 3.3.

- the observation that convergence of GPS algorithms for MVP problems requires the specific assumption that discrete neighbors of any iterate must lie on the current mesh.

- an improved construction of the mesh for MVP problems, in which a different generating matrix may be used for each set of discrete variable values, while preserving all the convergence properties of the algorithms.

- the new convergence results of Section 5.3 for the MGPS algorithm for bound and linear constrained MVP problems. These results apply weaker hypotheses

than those found in [8] to more fully characterize certain limit points, consistent with existing theory [6, 7].

- the FMGPS algorithm described in Chapter 6, with all of its development and convergence analysis given there.

- the weakening of the strict differentiability hypothesis needed for convergence to a first-order stationary point in constrained problems. This is applied throughout the results of Chapters 5 and 6.

- the implementation of the algorithms into the Matlab® software, NOMADm (see Section 7.1).

- A generalization of the thermal insulation system design problem described in Section 7.2, in which variable cross-sectional areas are permitted and nonlinear constraints on stress, mass, and thermal contraction are added.

- the successful study of numerical performance of the FMGPS algorithm on the thermal insulation system problem, which demonstrates the ability of FMGPS to solve mixed variable, nonlinearly constrained problems.

- much of the theory surrounding the GPS algorithm with derivative information; most notably, the theorems that describe how to prune the $\{-1, 0, 1\}^n$ positive spanning set to a singleton when the gradient or an approximation is available.

- the numerical experiments in Chapter 8, in which different approaches are tested on problems from the CUTE test set.

9.2   Future Areas of Research

From this work, several observations and questions have arisen that are now topics of future research and are now discussed.

Additional Study of the Filter Approach Perhaps the most important drawback of the filter GPS algorithm is its inability to ensure convergence to a first-order stationary point. Another is the additional hypotheses necessary to ensure that certain generalized derivatives are nonnegative.

To address the latter concern, one possible area of future work is to modify the filter algorithm so that the maximum allowable constraint violation $h_{\max}$ is allowed to decrease during the iteration sequence. A similar scheme is used by Fletcher *et al.* [49] to avoid the problem of blocking entries discussed in Section 4.2.1. The idea of blocking entries seems to be closely related to the additional hypothesis in Theorems 6.18 and 6.19; *i.e.*, that the current poll center (as opposed to a different filter point) must filter a trial point in a direction infinitely often in a refining subsequence, in order to ensure nonnegativity of the generalized directional derivative in that direction.

However, this modification does not appear to solve the problem of trying to converge to a KKT point if the functions are sufficiently smooth there. It is not clear that guaranteed convergence is even possible with a filter pattern search approach, since being able to generate the tangent cone at the limit point would typically require the algorithm to use an infinite number of directions. The grid-based methods of Coope and Price [34] offer an alternative construction that allows an infinite number of directions, but their construction requires that the directions satisfy a uniform linear independence condition. Still, grid-based methods appear to be a promising area for achieving first-order convergence for general NLP problems.

GPS with Derivative Information for Nonlinear Constrained Problems.
One of our major unexplored areas that still needs a lot of rigorous development is how to exploit derivative information in the presence of general nonlinear constraints. Our current implementation in NOMADm uses a variant of the feasible directions algorithm [139], in which a feasible descent direction is found and then modified so that the resulting descent direction is still feasible and puts the next trial point on the mesh. This appears to force convergence to a KKT point, although some informal attempts at solving a relatively simple problem resulted in rather slow and inaccurate convergence. At this point, convergence to a KKT point is still a conjecture that has not been rigorously established. It is also unclear what, if anything, can be said about the quality of the solution when only some derivatives are available.

Additional Nonsmooth Analysis. Although a hierarchy of GPS convergence results is now well established, we have not been able to establish conditions for which $0 \in \partial f(x)$ but $\nabla f(x)$ either does not exist or is nonzero at a point x. We have examples in which the gradient does not exist at x and $0 \notin \partial f(x)$, even though generalized directional derivatives are all nonnegative in a set of positive spanning directions. Of course, if we have strict differentiability at a point having nonnegative generalized directional derivatives in a set of positive spanning directions, then we have $0 = \nabla f(x) \in \partial f(x)$. We also have an example of a convex function for which GPS converges to a point with nonzero gradient. However, the open question is whether GPS can converge to a point x at which $0 \notin \partial f(x)$, when the function f is differentiable, but not strictly differentiable.

Probabilistic Analysis. Since GPS methods are directional based methods, there is little chance in practice of converging to a stationary point that is not a local minimum; however, that possibility does exist, and there are a number of examples that illustrate this. In fact, Audet [4] shows a pathological case in which GPS con-

verges to a global maximum (it starts there and never moves). However, it would be interesting to see if stronger optimality results could be proved in probability or almost surely, if some randomness was introduced in the selection of starting points and poll directions, in order to avoid the pathological cases.

This approach is different from Hart [68, 69, 70], who performs some probabilistic analysis of an evolutionary pattern search method. In his approach, randomness is part of the algorithm, and convergence in probability is to a first-order stationary point. This is actually a weaker result than traditional GPS methods (*e.g.*, under continuously differentiable f, $P(\liminf_k \|\nabla f(x_k)\| = 0) = 1$).

GPS with Finite Differences. Since gradients can be used to prune directions for GPS, there is a certain argument for applying finite difference approximations in an effort to accelerate convergence without requiring explicit computation of gradients. This is briefly explored in a slightly different context by Coope and Price [34]. Such an implementation would have to address the truncation error associated with the approximation and how that affects the ability of the method to guarantee a descent direction. Strategies dealing with how often to update the approximation would also have to be explored.

As an example, suppose that each POLL step uses the standard $2n$ directions, $[I, -I]$. Then a poll that does not find an improved mesh point will cost $2n$ function evaluations. However, we can compute a centered-difference approximation to the gradient at the current poll center using those $2n$ function evaluations. Of course, the accuracy of the approximation will depend on how fine the mesh is at that iteration. At the next iteration, after the mesh size is refined, we can simply poll in one descent direction, based on the approximate gradient; thus saving $2n-1$ function evaluations. The main drawback to this approach is the potential inaccuracy of the approximation. It is possible to generate a descent direction with respect to the approximate gradient

that is actually *not* a descent direction. Therefore, updating the approximation as the mesh gets finer, may be required to ensure convergence to a first-order stationary point.

The use of finite difference approximations is not limited to mesh local optimizers. Even if an improved mesh point is found, the function values of the new and old iterates generate a backward-difference approximation to the directional derivative at the new iterate in the successful direction.

Increasing Flexibility in the SEARCH **Step.** In many engineering applications with very expensive function evaluations, a typical SEARCH step consists of optimizing a less costly surrogate function and mapping the resulting optimal point (or several "close-to-optimal" points) to the mesh. However, the forcing of points to the mesh in order to satisfy GPS theory can often lead to a point which is not an improvement over the incumbent design. Since this scenario can prove costly, adding more flexibility in the SEARCH step is something that is worth pursuing.

Two approaches appear to be worth considering. First, despite the lack of supporting theory, we have observed in practice that success has been achieved using surrogates without mapping the resulting point to the mesh. We observe that one can conduct the SEARCH on a different, more refined mesh than what is used in the POLL step, even though the theory does not explicitly allow it. The reason for this is that if the SEARCH step finds an improved mesh point (or an unfiltered point), then POLL is not performed, and if SEARCH does not find an improved mesh point (or unfiltered point), then it can be viewed as never having been performed. There is a certain argument that if the SEARCH step is performed on an ultra-fine mesh – on the level of machine precision, then the computer's floating point arithmetic acts as the function that forces points onto the mesh. While noting its effectiveness in practice, we have been unable to rigorously prove that this works, and it remains an open question

for further study. One complication is the non-uniformity of machine-representable numbers.

The other approach, which is potentially more fruitful, is the grid-based methods of Coope and Price [35]. This method is similar to the basic GPS in that points are evaluated on a grid that is generated by a set of positive spanning directions. But unlike GPS methods, they allow the evaluation of points not on the grid. An attempt is made to move back to the grid in subsequent iterations, but in certain cases, the next grid is translated to the new point instead. The drawback to this approach is that infinite refinement of the mesh must be assumed; whereas, this is proved for GPS methods. Grid-based methods also require a sufficient decrease condition (*e.g.*, see [41]) to guarantee convergence; whereas GPS methods require only simple decrease in the objective function value.

As previously discussed, one other benefit of grid-based methods is that they allow grids to be constructed using an infinite set of directions, which can be beneficial in devising a derivative-free method that converges to a KKT point. However, this added flexibility generates a requirement that the directions satisfy a uniform linear independence condition. Serious implementation issues must be addressed, but the potential for solving nonlinearly constrained problems makes this an attractive research area.

Hybrid Algorithms. Finally, since the SEARCH step provides the user with great flexibility, it also allows the user to construct hybrid algorithms. The search heuristics mentioned in Section 2.2 are certainly among the possibilities for this option. For example, since the simulated annealing algorithm (see Section 2.2.1) can be guaranteed, under certain conditions, to converge in probability to the global optimum, it would be interesting to find out what can be proved if it were applied to a surrogate function within the SEARCH step of a GPS algorithm. It may be possible to construct

the hybrid in a way that would retain GPS theoretical properties and performance, but would also preserve the global quality of the solution that simulated annealing theory provides. This may be possible if each surrogate function can be proved to converge to its corresponding true function in the limit.

Computational Studies. There does not appear to be any significant computational studies that involve GPS methods – either comparisons of different implementations, or comparisons with other algorithms. In particular, a computational study of the augmented Lagrangian approach of Lewis and Torczon [95] and the filter GPS method of Audet and Dennis [7] would be very useful. While the former guarantees convergence to a KKT point for sufficiently smooth functions (which the latter does not), it requires the user to deal with a penalty parameter and derivative-free Lagrange multiplier estimates, which can significantly affect numerical performance and tends to be highly problem-dependent.

Test Problem Sets. Related to such studies is the need for a suite of engineering test problems similar to the classes of problems targeted here. Specifically, there are very few readily available test problems in which functions are expensive black box simulations with no limitations as to the local smoothness properties or variable types. A collection of such problems would be of great benefit to the optimization community in evaluating performance of GPS and other algorithms.

Other Software Issues. In addition to the areas described here, several improvements can be made with respect to the NOMADm software package. Perhaps the most important and helpful improvement would be to make it more adaptable to other software packages and codes. Currently, the only capability NOMADm has in this area is the ability to interface with pre-compiled or Matlab®-compiled FORTRAN-77 function files. It would be a much more attractive package if it could be linked with C, C++, and AMPL® [53] files.

Bibliography

[1] M. A. Abramson. Nonlinear optimization with mixed variables and derivatives–Matlab$^{\circledR}$ (NOMADm). Software. Available from Mark Abramson's NOMADm Page, http://www.caam.rice.edu/~abramson/NOMADm.html, 2002.

[2] M. A. Abramson, C. Audet, and J. E. Dennis, Jr. Generalized pattern searches with derivative information. Technical Report TR02-10, Department of Computational and Applied Mathematics, Rice University, Houston Texas, 2002.

[3] J. W. Atmar. *Speculation on the Evolution of Intelligence and its Possible Realization in Machine Form.* PhD thesis, New Mexico State University, Las Cruces, NM, 1976.

[4] C. Audet. Convergence results for pattern search algorithms are tight. Technical Report TR98-24, Department of Computational and Applied Mathematics, Rice University, Houston Texas, 1998. Also published as CRPC-TR98779.

[5] C. Audet, A. J. Booker, J. E. Dennis, Jr., P. D. Frank, and D. W. Moore. A surrogate-model-based method for constrained optimization. In *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, volume 2000-4891. AIAA, 2000.

[6] C. Audet and J. E. Dennis, Jr. Analysis of generalized pattern searches. Technical Report TR00-07, Department of Computational and Applied Mathematics, Rice University, Houston Texas, 2000. To appear in *SIAM Journal on Optimization*.

[7] C. Audet and J. E. Dennis, Jr. A pattern search filter method for nonlinear programming without derivatives. Technical Report TR00-09, Department of Computational and Applied Mathematics, Rice University, Houston Texas, 2000.

[8] C. Audet and J. E. Dennis, Jr. Pattern search algorithms for mixed variable programming. *SIAM Journal on Optimization*, 11(3):573–594, 2001.

[9] T. Bäck, F. Hoffmeister, and H.-P. Schwefel. A survey of evolution strategies. In R. Belew and L. Booker, editors, *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 2–9, Los Altos, CA, 1991. Morgan Kaufman.

[10] T. Bäck, G. Rudolph, and H.-P. Schwefel. Evolutionary programming and evolution strategies: Similarities and differences. In D. B. Fogel and J. W. Atmar, editors, *Proceedings of the Second Annual Conference on Evolutionary Programming*, pages 11–22, La Jolla, CA, 1993. Evolutionary Programming Society.

[11] J. E. Baker. *An Analysis of the Effects of Selection in Genetic Algorithms.* PhD thesis, Vanderbilt University, Nashville, Tennessee, 1989.

[12] S. Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta Mathematicae*, 3:133–181, 1922.

[13] R. F. Barron. *Cryogenic Systems*. McGraw-Hill, New York, 1966.

[14] M. S. Bazaraa and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley and Sons, Inc., New York, 1979.

[15] A. Bejan. A general variational principle for thermal insulation system design. *International Journal of Heat and Mass Transfer*, 22(2):219–228, February 1979.

[16] C. J. P. Bélisle. Convergence theorems for a class of simulated annealing algorithms on $\Re^d$. *Journal of Applied Probability*, 29(4):885–892, December 1992.

[17] G. L. Bilbro and W. E. Snyder. Optimization of functions with many minima. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(4):840–849, 1991.

[18] I. Bongartz, A. R. Conn, N. I. M. Gould, and P. L. Toint. CUTE: Constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software*, 21(1):123–160, March 1995.

[19] A. Booker, J. E. Dennis, Jr., P. Frank, D. Moore, and D. Serafini. Managing surrogate objectives to optimize a helicopter rotor design — further experiments. Technical Report 98-4717, AIAA, St. Louis, September 1999.

[20] A. Booker, J. E. Dennis, Jr., P. Frank, D. Serafini, and V. Torczon. Optimization using surrogate objectives on a helicopter test example. In J. Burns and E. Cliff, editors, *Optimal Design*, Philadelphia, 1998. SIAM.

[21] A. J. Booker, J. E. Dennis, Jr., P. D. Frank, D. B. Serafini, V. Torczon, and M. W. Trosset. A rigorous framework for optimization of expensive function by surrogates. *Structural Optimization*, 17(1):1–13, 1999.

[22] B. Borchers and J. E. Mitchell. An improved branch and bound algorithm for mixed integer nonlinear programs. *Computers and Operations Research*, 21(4):359–367, 1994.

[23] G. E. P. Box. Evolutionary operation: A method for increasing industrial productivity. *Applied Statistics*, 6(2):81–101, June 1957.

[24] M. J. Bünner, K. Schittkowski, and G. van de Braak. Optimal design of surface acoustic wave filters for signal processing by mixed-integer nonlinear programming. Technical report, University of Bayreuth, 2002.

[25] S. L. Burgee, A. A. Giunta, V. Balabanov, B. Grossman, W. H. Mason, R. Narducci, R. T. Haftka, and L. T. Watson. A coarse-grained parallel variable-complexity multidisciplinary optimization paradigm. *Internation Journal on Supercomputing Applications and High Performance Computing*, 10(4):269–299, 1996.

[26] G. H. Burgin. On playing two-person zero-sum games against npnminimax players. *IEEE Transactions on Systems, Man, and Cybernetics*, SSC-5(4):369–370, 1968.

[27] G. H. Burgin. Systems identification by quasilinear and evolutionary programming. *Journal of Cybernetics*, 3(2):56–75, 1973.

[28] R. H. Byrd and R. A. Tapia. An extension of Curry's thereom to steepest descent in normed linear spaces. *Mathematical Programming*, 9:247–254, 1975.

[29] V. Černý. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1):41–51, 1985.

[30] J. C. Chato and J. M. Khodadidi. Optimization of cooled shields in insulations. *ASME Transactions, Journal of Heat Transfer*, 106(4):871–875, November 1984.

[31] F. H. Clarke. *Optimization and Nonsmooth Analysis*. SIAM Classics in Applied Mathematics, 5. SIAM, 1990.

[32] A. R. Conn, N. I. M. Gould, and P. L. Toint. A globally convergent augmented Lagrangian algorithm for optimzation with general constraints and simple bounds. *SIAM Journal on Numerical Analysis*, 28(2):545–572, April 1991.

[33] I. D. Coope and C. J. Price. A direct search conjugate directions algorithm for unconstrained minimization. Technical Report 188, Dept of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand, November 1999.

[34] I. D. Coope and C. J. Price. Positive bases in numerical optimization. Technical Report UCDMS2000/12, Dept of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand, August 2000.

[35] I. D. Coope and C. J. Price. On the convergence of grid-based methods for unconstrained optimization. *SIAM Journal on Optimization*, 11(4):859–869, 2001.

[36] R. J. Dakin. A tree-search algorithm for mixed integer programming problems. *Computer Journal*, 8(3):250–255, October 1965.

[37] C. Darwin. *On the Origin of Species by Means of Natural Selection*. John Murray, London: Albemarle Street, 1859.

[38] C. Davis. Theory of positive linear dependence. *American Journal of Mathematics*, 76(4):733–746, October 1954.

[39] K. A. De Jong. *An Analysis of the Behavior of a Class of Genetic Adaptive Systems*. PhD thesis, University of Michigan, Ann Arbor, 1975.

[40] J. E. Dennis, Jr. Danish meeting on surrogate optimization. Technical report, Department of Computational and Applied Mathematics, Rice University, Houston Texas, 2000.

[41] J. E. Dennis, Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Classics in Applied Mathematics, 16. SIAM, 1996.

[42] J. E. Dennis, Jr. and V. Torczon. Direct search methods on parallel machines. *SIAM Journal on Optimization*, 1(4):448–474, 1991.

[43] M. A. Duran and I. E. Grossman. An outer-approximation algorithm for a class of mixed integer nonlinear programs. *Mathematical Programming*, 36(3):306–339, December 1986.

[44] T. F. Durham, R. M. McClintock, and R. P. Reed. *Cryogenic Materials Data Handbook.* Office of Technical Services, Washington, DC, 1962.

[45] G. Ferlin, B. Jenninger, P. Lebrun, G. Peon, G. Riddone, and B. Szeless. Comparison of floating and thermalized multilayer insulation systems at low boundary temperature. Technical Report 21, Large Hadron Collider Project, 1996.

[46] R. Fletcher and S. Leyffer. Solving mixed integer programs by outer approximation. *Mathematical Programming*, 66(3):327–349, September 1994.

[47] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, 91(2):239–269, January 2002.

[48] R. Fletcher, S. Leyffer, and P. L. Toint. On the global convergence of an SLP–filter algorithm. Technical Report NA/183, Dundee University, Department of Mathematics, 1998.

[49] R. Fletcher, S. Leyffer, and P. L. Toint. On the global convergence of an SQP–filter algorithm. *SIAM Journal on Optimization*, 13(1):44–59, 2002.

[50] C. A. Floudas. *Nonlinear and Mixed Integer Optimization.* Oxford University Press, 1995.

[51] L. J. Fogel, A. J. Owens, and M. J. Walsh. *Artificial Intelligence Through Simulated Evolution.* John Wiley & Sons, New York, 1966.

[52] S. Forrest and M. Mitchell. What makes a problem hard for a genetic algorithm? Some anomolous results and their explanation. *Machine Learning*, 13(2/3), 1993.

[53] R. Fourer, D. M. Gay, and B. W. Kernighan. *AMPL: A Modelling Language for Mathematical Programming.* Duxbury Press, Belmont, CA, 1993.

[54] S. B. Gelfand and S. K. Mitter. Metropolis-type annealing algorithms for global optimization. *SIAM Journal on Control and Optimization*, 31(1):111–131, January 1993.

[55] S. Geman and D. Geman. Stochastic relaxation, Gibbs distribution, and the Bayesian restoration in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(6):721–741, November 1984.

[56] A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260, 1972.

[57] B. Gidas. Nonstationary Markov chains and convergence of the annealing algorithm. *Journal of Statistical Physics*, 39(1/2):73–131, 1985.

[58] A. A. Giunta. *Aircraft Multidisciplinary Optimization Using Design of Experiments Theory and Response Surface Modeling Methods*. PhD thesis, Department of Aerospace and Ocean Engineering, Virginia Polytechnic Institute and State University, May 1997.

[59] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5):533–549, 1986.

[60] F. Glover. Tabu search–part I. *ORSA Journal on Computing*, 1(3):190–206, Summer 1989.

[61] F. Glover. Tabu search–part II. *ORSA Journal on Computing*, 2(1):4–32, Winter 1990.

[62] F. Glover. Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). *Discrete Applied Mathematics*, 49(1-3):231–255, 1994.

[63] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, New York, 1989.

[64] D. E. Goldberg and P. Segrest. Finite markov chain analysis of genetic algorithms. In J. J. Grefensette, editor, *Proceedings of the Second International Conference on Genetic Algorithms*, pages 1–8, Hillsdale, NJ, 1987. Lawrence Erlbaum Associates.

[65] O. K. Gupta and A. Ravindran. Branch and bound experiments in convex nonlinear integer programming. *Management Science*, 31(12):1533–1546, December 1985.

[66] B. Hajek. Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13(2):311–329, 1988.

[67] P. Hansen. The steepest ascent mildest descent heuristic for combinatorial programming. In *Conference on Numerical Methods in Combinatorial Optimisation*, Capri, Italy, 1986.

[68] W. E. Hart. A generalized stationary point convergence theory for evolutionary algorithms. In *Proceddings of the International Conference on Genetic Algorithms*, pages 127–134, 1997.

[69] W. E. Hart. A stationary point convergence theory for evolutionary algorithms. In *Foundations of Genetic Algorithms, 4*, pages 325–342, 1997.

[70] W. E. Hart. A convergence analysis of unconstrained and bound constrained evolutionary pattern search. *Evolutionary Computation*, 2000. To appear.

[71] M. A. Hilal and R. W. Boom. Optimization of mechanical supports for large superconductive magnets. In K. D. Timmerhaus, R. P. Reed, and A. F. Clark, editors, *Advances in Cryogenic Engineering*, volume 22, pages 224–232. Plenum Press, New York, 1977.

[72] M. A. Hilal and Y. M. Eyssa. Minimization of refrigeration power for large cryogenic systems. In K. D. Timmerhaus and H. A. Snyder, editors, *Advances in Cryogenic Engineering*, volume 25, pages 350–357. Plenum Press, New York, 1980.

[73] J.-B. Hiriart-Urruty and C. Lemaréchal. *Convex Analysis and Minimization Algorithms*. Springer-Verlag, Berlin, 1993.

[74] J. H. Holland. Outline for a logical theory of adaptive systems. *Journal of the Association of Computing Machinery*, 3:297–314, 1962.

[75] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, Michigan, 1975.

[76] R. Hooke and T. A. Jeeves. Direct search solution of numerical and statistical problems. *Journal of the Association of Computing Machinery*, 8(2):212–229, 1961.

[77] P. Hough, T. G. Kolda, and V. Torczon. Asynchronous parallel pattern search for nonlinear optimization. Technical Report SAND2000-8213, Sandia National Laboratories, 2000.

[78] L. Ingber. Very fast simulated re-annealing. *Mathematical Computer Modelling*, 12(8):967–973, 1989.

[79] L. Ingber. Simulated annealing: Practice versus theory. *Mathematical Computer Modelling*, 18(11):29–57, 1993.

[80] L. Ingber and B. Rosen. Genetic algorithms and very fast simulated reannealing: A comparison. *Mathematical Computer Modelling*, 16(11):87–100, 1992.

[81] B. Jenninger, G. Peon, and B. Szeless. Characterization of net type thermal insulators at 1.8 k low boundary temperature. Technical report, Large Hadron Collider Project, 1996.

[82] F. John. Extremum problems with inequalities as subsidiary conditions. In K. O. Friedrichs, O. E. Neugebauer, and J. J. Stoker, editors, *Studies and Essays: Courant Anniversary Volume*, pages 187–204. Wiley-Interscience, New York, 1948.

[83] W. Karush. Minima of functions of several variables with inequalities as side conditions. Master's thesis, Department of Mathematics, University of Chicago, 1939.

[84] J. E. Kelley. The cutting plane method for solving convex programs. *Journal of SIAM*, 8(4):703–712, December 1960.

[85] D. R. Kincaid and E. W. Cheney. *Numerical Analysis.* Brooks/Cole Publishing Company, Pacific Grove, California, 2nd edition, 1996.

[86] J. Kingdon. Genetic algorithms: Deception, convergence and starting conditions. Technical report, Department of Computer Science, University College, London, 1992.

[87] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[88] G. R. Kocis and I. E. Grossman. Relaxation strategy for the structural optimization process flow sheets. *Industrial and Engineering Chemistry Research*, 26(9):1869–1879, 1987.

[89] M. Kokkolaras, C. Audet, and J. E. Dennis, Jr. Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system. *Engineering and Optimization*, 2(1):5–29, 2001.

[90] T. G. Kolda and V. Torczon. On the convergence of asynchronous parallel pattern search. Technical Report SAND2001-8696, Sandia National Laboratories, December 2001.

[91] T. G. Kolda and V. Torczon. Understanding asynchronous parallel pattern search. Technical Report SAND2001-8695, Sandia National Laboratories, December 2001.

[92] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In J. Neyman, editor, *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, CA, 1951. Univeristy of California Press.

[93] R. M. Lewis and V. Torczon. Rank ordering and positive basis in pattern search algorithms. Technical Report TR 96-71, ICASE NASA Langley Research Center, 1996.

[94] R. M. Lewis and V. Torczon. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, 9(4):1082–1099, 1999.

[95] R. M. Lewis and V. Torczon. Pattern search algorithms for linearly constrained minimization. *SIAM Journal on Optimization*, 10(3):917–941, 2000.

[96] R. M. Lewis and V. Torczon. A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. *SIAM Journal on Optimization*, 12(4):1075–1089, 2002.

[97] S. Leyffer. Generalized outer approximation. Technical Report NA/178, Dundee University, Department of Mathematics, 1997. To appear in *Encyclopedia of Optimization*, editors C. M. Floudas and P.I.M. Pardalos.

[98] S. Leyffer. Integrating SQP and branch-and-bound for mixed integer nonlinear programming. Technical Report NA/182, Dundee University, Department of Mathematics, 1998.

[99] Q. Li, X. Li, G. E. McIntosh, and R. W. Boom. Minimization of total refrigeration power of liquid neon and nitrogen cooled intercepts for SMES magnets. In R. W. Fast, editor, *Advances in Cryogenic Engineering*, volume 35, Part A, pages 833–840. Plenum Press, New York, 1989.

[100] M. Locatelli. Convergence properties of simulated annealing algorithms for continuous global optimization. *Journal of Applied Probability*, 33(4):1127–1140, 1996.

[101] M. Locatelli. Simulated annealing algorithms for continuous global optimization: Convergence conditions. *Journal of Optimization Theory and Applications*, 104(1):121–133, January 2000.

[102] M. Lundy and A. Mees. Convergence of an annealing algorithm. *Mathematical Programming*, 34(1):111–124, January 1986.

[103] M. M. Mäkelä and P. Neittaanmäki. *Nonsmooth Optimization*. World Scientific, Singapore, 1992.

[104] O. L. Mangasarian. *Nonlinear Programming*. McGraw-Hill, Inc., New York, 1969.

[105] M. Mathieu, V. Parma, T. Renaglia, P. Rohmig, and L. R. Williams. 293 k – 1.9 k supporting systems for the large hadron collider (lhc) cryo-magnets. Technical Report 163, Large Hadron Collider Project, 1998.

[106] R. M. McClintock and H. P. Gibbons. *Mechanical Properties of Structural Materials at Low Temperatures*. Washington, DC, June 1960.

[107] M. D. McKay, W. J. Conover, and R. J. Beckman. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[108] Metals and Ceramics Information Center, Batelle, Columbus Laboratories. *Handbook on Materials for Superconducting Machinery: Mechanical, Thermal, Electrical, and Magnetic Properties of Structural Materials*, 1974.

[109] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, June 1953.

[110] Z. Michalewicz. *Genetic Algorithms + Data Structure = Evolution Programs*. Springer-Verlag, Berlin, 3rd edition, 1994.

[111] D. Mitra, F. Romeo, and A. L. Sangiovanni-Vincentelli. Convergence and finite-time behavior of simulated annealing. *Advanced Applied Probability*, 18(3):747–771, September 1986.

[112] Z. Musicki, M. A. Hilal, and G. E. McIntosh. Optimization of cryogenic and heat removal system of space borne magnets. In R. W. Fast, editor, *Advances in Cryogenic Engineering*, volume 35, Part B, pages 975–982. Plenum Press, New York, 1989.

[113] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7(4):308–313, January 1965.

[114] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer Series in Operations Research. Springer-Verlag, New York, 1999.

[115] R. H. J. M. Otten and L. P. P. P. van Ginneken. Floorplan design using simulated annealing. In *Proceedings of IEEE International Conference on Computer-Aided Design*, pages 96–98, Santa Clara, 1984.

[116] A. B. Owen. Orthogonal arrays for computer experiments, integration, and visualization. *Statistica Sinica*, 2:439–452, 1992.

[117] M. Pincus. A monte carlo method for the approximate solution of certian types of constrained optimization problems. *Operations Research*, 18(6):1225–1228, November–December 1970.

[118] I. Quesada and I. E. Grossman. An LP/NLP based branch and bound algorithm for convex MINLP optimization problems. *Computers and Chemical Engineering*, 16(10/11):937–947, 1992.

[119] R. T. Rockafeller. *Convex Analysis*. Princeton University Press, Princeton, NJ, 1970.

[120] G. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101, 1994.

[121] R. A. Runtenbar. Simulated annealing algorithms: An overview. *IEEE Circuits and Devices Magazine*, 5(1):19–26, January 1989.

[122] M. Russo and L. C. Jain. An introduction to evolutionary computing. In L. C. Jain, editor, *Evolution of Engineering and Information Systems and Their Applications*, International Series on Computational Intelligence, chapter 1, pages 3–29. CRC Press, Boca Raton, 2000.

[123] H.-P. Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Chichester, UK, 1981.

[124] H.-P. Schwefel. *Evolution and Optimum Seeking*. John Wiley & Sons, New York, 1995.

[125] J. E. Shigley and L. D. Mitchell. *Mechanical Engineering Design*. McGraw-Hill, New York, 4th edition, 1983.

[126] M. Stein. Large sample properties of simulations using latin hypercube sampling. *Technometrics*, 29(2):143–151, 1987.

[127] H. Szu and R. Hartley. Fast simulated annealing. *Physics Letters A*, 122(3/4):157–162, 1987.

[128] R. Tanese. *Distributed Genetic Algorithms for Function Optimization*. PhD thesis, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, 1989.

[129] B. Tang. Orthogonal array-based latin hypercubes. *Journal of the American Statistical Association*, 88(424):1392–1397, 1993.

[130] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25, 1997.

[131] M. W. Trosset. What is simulated annealing? *Optimization and Engineering*, 2(2):201–213, June 2002.

[132] M. W. Trosset and V. Torczon. Numerical optimization using computer experiments. Technical Report 97-38, ICASE NASA Langley Research Center, 1997.

[133] H. Tuy. *Convex Analysis and Global Optimization*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1998.

[134] D. Vanderbilt and S. G. Louie. A monte carlo simulated annealing approach to optimization over continuous variables. *Journal of Computational Physics*, 56(2):259–271, August 1984.

[135] J. Viswanathan and I. E. Grossman. A combined penalty function and outer approximation for MINLP optimization. *Computers and Chemical Engineering*, 14(7):769–782, 1990.

[136] T. Westerlund and F. Pettersson. A cutting plane method for solving convex MINLP problems. *Computers and Chemical Engineering*, 19(Suppl.):S131–S136, 1995.

[137] M. Yamaguchi, T. Ohmori, and A. Yamamoto. Design optimization of a vapor-cooled radiation shield for LHe cryostat in space use. In R. W. Fast, editor, *Advances in Cryogenic Engineering*, volume 37, Part B, pages 1367–1375. Plenum Press, New York, 1991.

[138] X. Yao. Introduction. In X. Yao, editor, *Evolutionary Computation*, chapter 1, pages 1–36. World Scientific, Singapore, 1999.

[139] G. Zoutendijk. *Methods of Feasible Directions*. Elsevier, Amsterdam, 1960.

Appendix A

Test Problems

The purpose of this chapter is to provide the reader with additional data with respect to the computational results recorded in this document. Specifically, we now provide a mathematical description of each of the 18 CUTE [18] test problems referred to in Chapter 8 (see Section 8.5) and solved using the NOMADm software. The other problem addressed numerically in this document (*i.e.*, the thermal insulation system design problem) is described in detail in Section 7.2. Matlab® source code for all of these problems, along with the entire NOMADm source code, is available for download on the internet [1].

ALLINITU:

$$
\begin{aligned}
\min f(x_1, x_2, x_3, x_4) = \quad & x_3 - 1 + x_1^2 + x_2^2 + (x_3 + x_4)^2 + \sin(x_4)^4 \\
& + 2\sin(x_3)^2 + x_1^2 x_2^2 + x_4 - 3 + (x_4 - 1)^2 + x_2^4 \\
& + [x_3^2 + (x_1 + x_4)^2]^2 + [x_1 - 4 + \sin(x_4)^2 + x_2^2 x_3^2]^2
\end{aligned}
$$

Initial Point: not specified (used $(x_1, x_2, x_3, x_4) = (0, 1, 0.5, 2)$.)

ALLINIT:

$$\min f(x_1, x_2, x_3, x_4) = \quad x_3 - 1 + x_1^2 + x_2^2 + (x_3 + x_4)^2 + \sin(x_4)^4$$

$$+ 2\sin(x_3)^2 + x_1^2 x_2^2 + x_4 - 3 + (x_4 - 1)^2 + x_2^4$$

$$+ [x_3^2 + (x_1 + x_4)^2]^2 + [x_1 - 4 + \sin(x_4)^2 + x_2^2 x_3^2]^2$$

subject to

$$0 \leq x_2$$

$$-10^{10} \leq x_3 \leq 1$$

$$2 = x_4$$

Initial Point: not specified (used $(x_1, x_2, x_3, x_4) = (0, 1, 0.5, 2)$.)

BARD:

$$\min f(x_1, x_2, x_3) = \sum_{j=1}^{16} \left[Y_j - \left(x_1 + \frac{j}{x_2(16 - j) + x_3 \min\{j, 16 - j\}} \right) \right]^2,$$

where $Y = (.14, .18, .22, .25, .29, .32, .35, .39, .37, .58, .73, .96, 1.34, 2.10, 4.39)$.

Initial Point: $(x_1, x_2, x_3) = (1, 1, 1)$

BOX2:

$$\min f(x_1, x_2, x_3) = \sum_{j=1}^{10} \left(e^{-0.1jx_1} - e^{-0.1jx_2} - x_3 e^{-0.1j} + x_3 e^{-j} \right)^2$$

subject to $x_3 = 1$

Initial Point: $(x_1, x_2, x_3) = (0, 10, 1)$

BOX3:

$$\min f(x_1, x_2, x_3) = \sum_{j=1}^{10} \left(e^{-0.1jx_1} - e^{-0.1jx_2} - x_3 e^{-0.1j} + x_3 e^{-j} \right)^2$$

Initial Point: $(x_1, x_2, x_3) = (0, 10, 1)$

DENSCHNA:

$$\min f(x_1, x_2) = x_1^4 + (x_1 + x_2)^2 + (e^{x_2} - 1)^2$$

Initial Point: $(x_1, x_2) = (1, 1)$

DENSCHNB:

$$\min f(x_1, x_2) = (x_1 - 2)^2 + [x_2(x_1 - 2)]^2 + (x_2 + 1)^2$$

Initial Point: $(x_1, x_2) = (1, 1)$

DENSCHNC:

$$\min f(x_1, x_2) = (x_1^2 + x_2^2 - 2)^2 + [e^{(x_1 - 1)} + x_2^3 - 2]^2$$

Initial Point: $(x_1, x_2) = (2, 3)$

EXPFIT:

$$\min f(x_1, x_2) = \sum_{j=1}^{10} \left(x_1 e^{0.25 j x_2} - 0.25j \right)^2$$

Initial Point: $(x_1, x_2) = (1, 1)$

MARATOSB:

$$\min f(x_1, x_2) = x_1 + 10^{-6}(x_1^2 + x_2^2 - 1)^2$$

Initial Point: $(x_1, x_2) = (0, 0)$

MDHOLE:

$$\min f(x_1, x_2) = x_1 + 10^{-2}(\sin(x_1) - x_2)^2$$

$$\text{subject to } x_1 \geq 0$$

Initial Point: $(x_1, x_2) = (10, 10)$

MEXHAT:

$$\min f(x_1, x_2) = -2(x_1 - 1)^2 + 10^4(-0.02 + 10^{-4}(x_2 - x_1^2)^2 + (x_1 - 1)^2)^2$$

Initial Point: $(x_1, x_2) = (0.86, 0.72)$

MEYER3:

$$\min f(x_1, x_2, x_3) = \sum_{j=1}^{16} \left[Y_j - x_1 \exp\left(\frac{x_2}{x_3 + 45 + 5j} \right) \right]^2,$$

where $Y = \quad (34780, 28610, 23650, 19630, 16370, 13720, 11540, 9744, ...$

$8261, 7030, 6005, 5147, 4427, 3820, 3307, 2872).$

Initial Point: $(x_1, x_2, x_3) = (0.02, 4000, 250)$

OSBORNEA:

$$\min f(x_1, x_2, x_3, x4, x5) = \sum_{j=1}^{33} \left[Y_j - \left(x_1 + x_2 e^{(-10(j-1)x_4)} + x_3 e^{(-10(j-1)x_5)} \right) \right]^2,$$

where

$Y = \quad (0.844, 0.908, 0.932, 0.936, 0.925, 0.908, 0.881, 0.850, 0.818, 0.784, 0.751, ...$

$0.718, 0.685, 0.658, 0.628, 0.603, 0.580, 0.558, 0.538, 0.522, 0.506, 0.490, ...$

$0.478, 0.467, 0.457, 0.448, 0.438, 0.431, 0.424, 0.420, 0.414, 0.411, 0.406).$

Initial Point: $(x_1, x_2, x_3, x_4, x_5) = (0.5, 1.5, -1, 0.01, 0.02)$

OSBORNEB:

$$\min f(x_1, \ldots, x_{11}) = \sum_{j=1}^{75} [Y_j - (A_j + B_j + C_j + D_j)]^2$$

where

$$A_j = x_1 e^{(-.1(j-1)x_5)},$$

$$B_j = x_2 e^{-x_6(-.1(j-1)-x_9)^2},$$

$$C_j = x_3 e^{-x_7(-.1(j-1)-x_{10})^2},$$

$$D_j = x_4 e^{-x_8(-.1(j-1)-x_{11})^2}, \text{ and}$$

$$
\begin{aligned}
Y = \quad & (1.366, 1.191, 1.112, 1.013, 0.991, 0.885, 0.831, 0.847, 0.786, 0.725, \ldots \\
& 0.746, 0.679, 0.608, 0.655, 0.616, 0.606, 0.602, 0.626, 0.651, 0.724, \ldots \\
& 0.649, 0.649, 0.694, 0.644, 0.624, 0.661, 0.612, 0.558, 0.533, 0.495, \ldots \\
& 0.500, 0.423, 0.395, 0.375, 0.372, 0.391, 0.396, 0.405, 0.428, 0.429, \ldots \\
& 0.523, 0.562, 0.607, 0.653, 0.672, 0.708, 0.633, 0.668, 0.645, 0.632, \ldots \\
& 0.591, 0.559, 0.597, 0.625, 0.739, 0.710, 0.729, 0.720, 0.636, 0.581, \ldots \\
& 0.428, 0.292, 0.162, 0.098, 0.054).
\end{aligned}
$$

Initial Point: $(x_1, \ldots, x_{11}) = (1.3, 0.65, 0.65, 0.7, 0.6, 3, 5, 7, 2, 4.5, 5.5)$

<u>**OSLBQP:**</u>

$$\min f(x_1, \ldots, x_8) = x_1 + 2x_5 - x_8 + \frac{1}{2} \sum_{j=1}^{8} x_j^2$$

subject to

$$2.5 \ \leq \ x_1$$

$$0 \ \leq \ x_2 \ \leq 4.1$$

$$0 \ \leq \ x_3$$

$$0 \ \leq \ x_4$$

$$0.5 \ \leq \ x_5 \ \leq 4.0$$

$$0 \ \leq \ x_6$$

$$0 \ \leq \ x_7$$

$$0 \ \leq \ x_8 \ \leq 4.3$$

Initial Point: $(x_1, \ldots, x_8) = (0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5)$

<u>PALMER1</u>:

$$\min f(x_1, x_2, x_3, x_4) = \sum_{j=1}^{31} \left[Y_j - \left(x_1 X_j^2 + \frac{x_2}{x_3 + \frac{1}{x_4} X_j^2} \right) \right]^2$$

subject to $\quad x_i \geq 10^{-4}, \quad i = 2, 3, 4,$

where

$$X = \quad (-1.788963, -1.745329, -1.658063, -1.570796, -1.483530, -1.396263, \dots$$

$$-1.308997, -1.218612, -1.134464, -1.047198, -0.872665, -0.698132, \dots$$

$$-0.523599, -0.349066, -0.174533, 0.000000, 1.788963, 1.745329, \dots$$

$$1.658063, 1.570796, 1.483530, 1.396263, 1.308997, 1.218612, \dots$$

$$1.134464, 1.047198, 0.872665, 0.698132, 0.523599, 0.349066, 0.174533),$$

$$Y = \quad (78.596218, 65.77963, 43.96947, 27.038816, 14.6126, 6.2614, \dots$$

$$1.538330, 0.00000, 1.188045, 4.6841, 16.9321, 33.6988, \dots$$

$$52.3664, 70.1630, 83.4221, 88.3995, 78.596218, 65.77963, \dots$$

$$43.96947, 27.038816, 14.6126, 6.2614, 1.538330, 0.00000, \dots$$

$$1.188045, 4.6841, 16.9321, 33.6988, 52.3664, 70.1630, 83.4221).$$

Initial Point: $(x_1, x_2, x_3, x_4) = (1, 1, 1, 1)$

PALMER1A:

$$\min f(x_1, \dots, x_6) = \sum_{j=1}^{31} \left[Y_j - \left(x_1 + x_2 X_j^2 + x_3 X_j^4 + x_4 X_j^6 + \frac{x_5}{x_6 + X_j^2} \right) \right]^2$$

subject to $\quad x_i \geq 0, \quad i = 5, 6,$

where X and Y are defined in problem PALMER1.

Initial Point: $(x_1, x_2, x_3, x_4, x_5, x_6) = (1, 1, 1, 1, 1, 1)$

PALMER1B:

$$\min f(x_1, x_2, x_3, x_4) = \sum_{j=1}^{31} \left[Y_j - \left(x_1 X_j^2 + x_2 X_j^4 + \frac{x_3}{x_4 + X_j^2} \right) \right]^2$$

subject to $\quad x_i \geq 0, \quad i = 3, 4,$

where X and Y are defined in problem PALMER1.

Initial Point: $(x_1, x_2, x_3, x_4) = (1, 1, 1, 1)$

PALMER1C:

$$\min f(x_1, \ldots, x_8) = \sum_{j=1}^{31} \left[Y_j - \left(\sum_{k=1}^{8} x_k X_j^{2(k-1)} \right) \right]^2,$$

where X and Y are defined in problem PALMER1.

Initial Point: $(x_1, \ldots, x_8) = (1, 1, 1, 1, 1, 1, 1, 1)$