

## 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 unrec-

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 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.

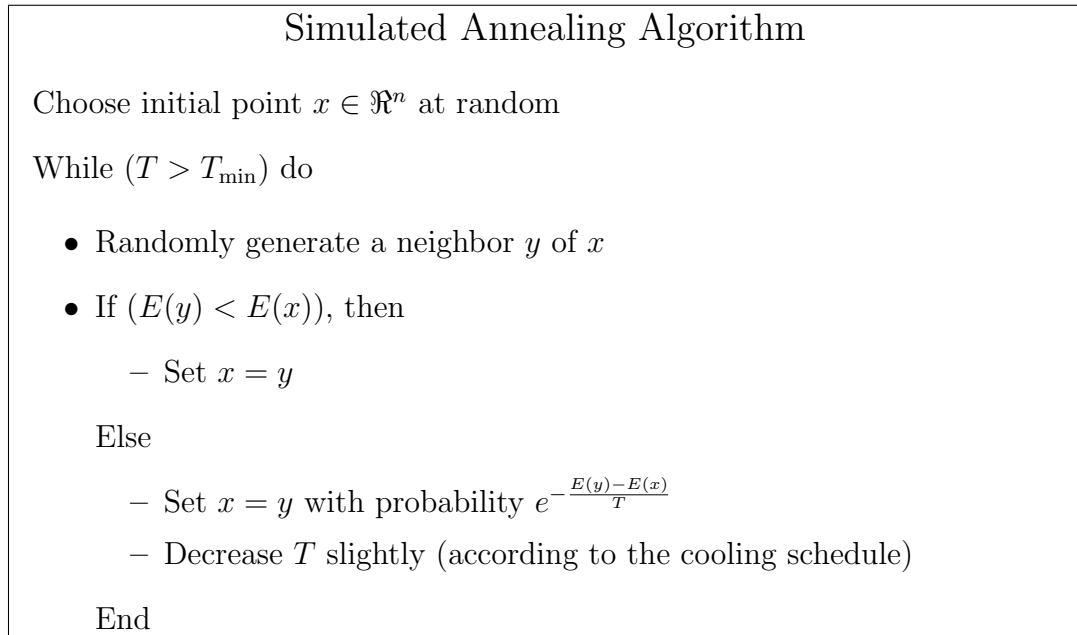
## 2.2 Search Heuristics

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  $E : \mathfrak{R}^n \rightarrow \mathfrak{R}$  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-



**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  $\mathfrak{R}^n$ . 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.

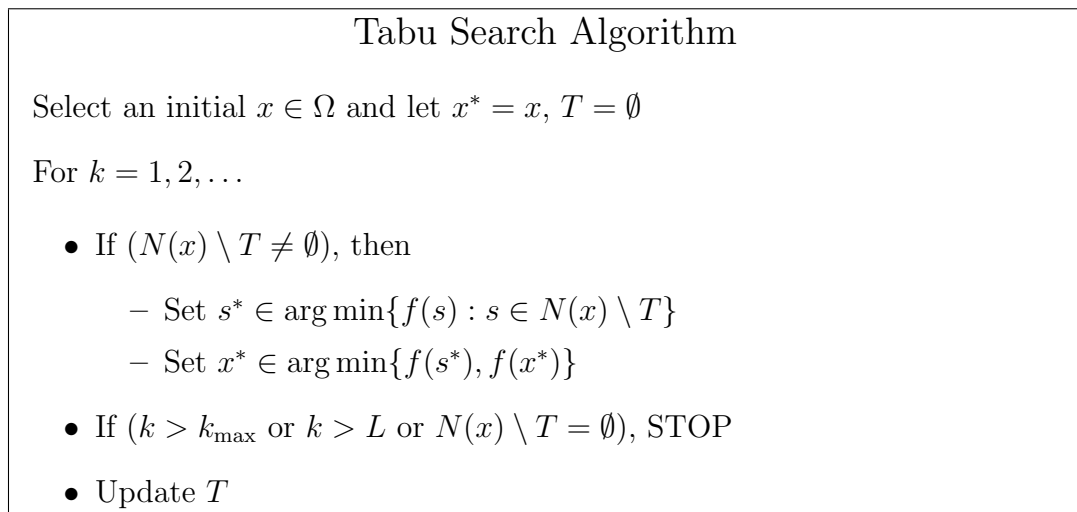
Ingber [78] offers some algorithmic improvements without sacrificing theoretical convergence. His implementation shows favorable performance results compared to that of the traditional simulated annealing [79] and genetic algorithms [80].

### **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.



**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:

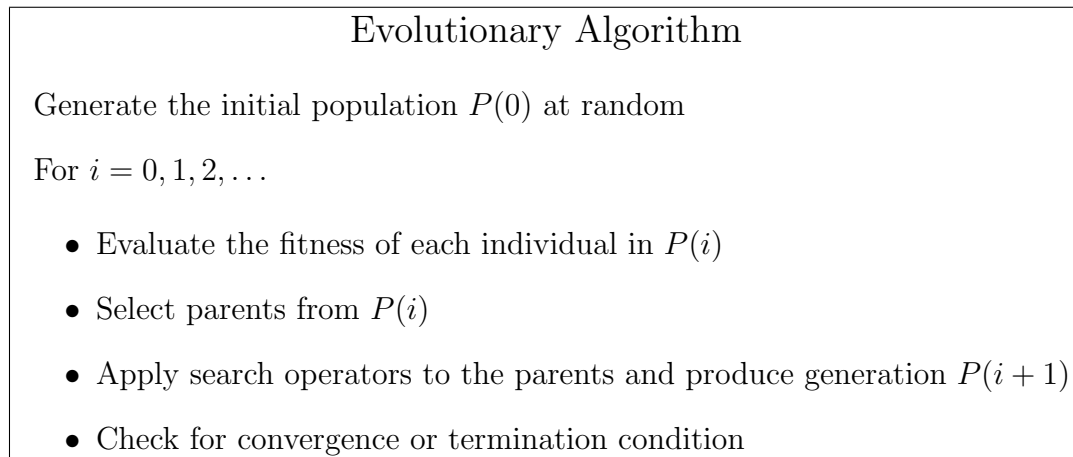
- **Selection:** Process by which parents are selected for reproduction.
- **Reproduction:** Process by which genes are passed from parents to offspring.
- **Mutation:** Random errors occurring in the reproduction process.
- **Competition:** Process by which offspring survive to the next generation when there are limited resources.

Given these definitions, it is helpful to point out a few examples. Suppose parents  $x = (x_1, x_2, \dots, x_n)$  and  $y = (y_1, y_2, \dots, y_n)$  have been selected. One common recombination operator produces two complementary offspring  $z, w \in \mathfrak{R}^n$ , 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



**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 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  $(x, \sigma)$ , where  $x$  is its position in the search space, and  $\sigma$  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  $\sigma$ . Since the selection of  $\sigma$  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  $\sigma$  as well as  $x$ .

The two primary selection operators are referred to as  $(\lambda + \mu)$  and  $(\lambda, \mu)$ , where  $\mu$  represents the number of parents and  $\lambda$  represents the number of offspring. After the  $\lambda$  offspring are generated, the  $\mu$  fittest individuals are selected, either from the total  $\mu + \lambda$  candidates or only the  $\lambda$  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  $\sigma$  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  $(x, \sigma)$  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  $\mu$  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  $\mu$  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  $\{x \in \mathfrak{R}^n : f(x) \leq f(x_0)\}$ , where  $x_0 \in \mathfrak{R}^n$  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  $\{x \in \mathfrak{R}^n : f(x) \leq f(x_0)\}$ , 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

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

where  $\lambda = (\lambda_1, \dots, \lambda_p)^T$  is the vector of Lagrange multiplier estimates for the equality constraints  $C_i(x), i = 1, 2, \dots, p$  (inequality constraints are assumed to have had slack variables added, as appropriate),  $\mu$  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  $\delta_j \rightarrow 0$  and  $\delta_0 \ll 1$ . The Lagrange multiplier estimates are updated by the same Hestenes-Powell formula as in [32]; namely,

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

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  $s_f : X \rightarrow \Re$  and  $s_C : X \rightarrow \Re^p$  are constructed based either on simplified physics, or approximations obtained by evaluating  $f$  and  $C$  at selected points, called *data sites*,  $x_1, \dots, x_r$ , 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, Büchner, 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.