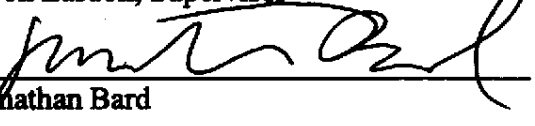
The Dissertation Committee for Zsolt Gyula Ugray certifies that this is the approved version of the following dissertation:

OQGRG: A Multi-Start Algorithm for Global Solution of

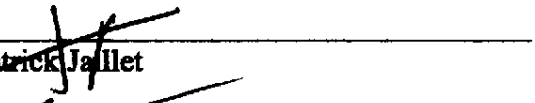
Nonlinear and Mixed Integer Programs

Committee:

Leon Lasdon, Supervisor

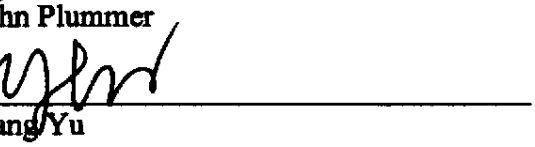


Jonathan Bard



Patrick Jaillet

John Plummer



Gang Yu

OQGRG: A Multi-Start Algorithm for Global Solution of Nonlinear and Mixed Integer Programs

by

Zsolt Gyula Ugray, MSEE

Dissertation

Presented to the Faculty of the Graduate School of

the University of Texas at Austin

In Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

December, 2001

.

ProQuest Information and Learning Company
300 North Zeeb Road
PO Box 1346
Ann Arbor, MI 48106-1346

I’d like to dedicate this work to my parents and grandparents who have always believed in life-long learning and who have taught me that only hard work will bring good results.

.

Acknowledgements

Looking back at the past several years leading to this dissertation I cannot help but wonder if I could have gotten here without all the trust, support, and encouragement of so many people around me. I'm grateful to Sherry, my wife, with whom I've shared the joys and troubles of every day for the last few years, and who has always showed love, happiness, and affection. My mother, father, and brother, Csilla, László, and Gábor Ugray, have supported me in every imaginable way, every day, despite the thousands of miles we had to live apart. There is not enough space here to list all my family and numerous good friends from Austin, from my home country, and from all over the world, who cheered me up with their high spirits and gave me encouragement, motivation, often unknown even to themselves.

I'm greatly indebted to my Supervisor, Leon Lasdon, who not only patiently guided me through the process of this research, but who also gave me the opportunity to learn a lot from him, not only academically. Many thanks are due to my committee members, Patrick Jaillet, Gang Yu, and Jonathan Bard for their suggestions and their availability for advice. Special thanks are due to John Plummer, whose resources are limitless when it comes to coding, as proved by the GAMS interface.

I'd also like to thank Fred Glover, Manuel Laguna, and Jim Kelly of the University of Colorado, Boulder, for their assistance in letting me use OptQuest, their Scatter Search implementation, as a component of OQGRG. Listening to their experiences helped make the algorithm better.

Again, there is not enough room to mention everyone at UT Austin, in the Graduate School of Business and in the MSIS Department, who made my experience here so wonderful, exciting, and intellectually stimulating. They gave me many opportunities for growth as a scholar and as a person.

.

OQGRG: A Multi-Start Algorithm for Global Solution of Nonlinear and Mixed Integer Programs

Publication No._____

Zsolt Gyula Ugray, Ph.D.
The University of Texas at Austin, 2001

Supervisor: Leon Lasdon

Economical, managerial, engineering, and natural systems are often represented by nonlinear equations and inequalities, using discrete and continuous variables. Global Optimization provides methodologies to find the global solutions for the prescriptive models that attempt to describe, predict, and optimize their behavior. OQGRG, the algorithm presented in this dissertation, was developed to solve problems in this large target class of mixed integer, nonlinear, constrained optimization models that often have multiple local optima. OQGRG is a multi-start, 2-stage, global optimization algorithm that combines the efficiency of the Scatter Search meta-heuristic and the power of a reduced gradient nonlinear solver. It uses OptQuest as the implementation of Scatter Search and Lsgrg2 as a nonlinear local solver. OQGRG is written in standard ANSI C, and a GAMS interface provides access to many test problems available in the literature. The effectiveness of the algorithm is demonstrated by solving 155 of 159 test problems within 1% of their best known solutions.

.

Table of Contents

.

1. Introduction

Physical, chemical, biological, economical, engineering, and managerial systems can often be described by nonlinear equations and inequalities, using many variables, some of them discrete. It is not surprising, that prescriptive models that attempt to describe, predict, and optimize their behavior lead to constrained nonlinear decision models that often have discrete variables as well. Many different approaches have been suggested for solving these complex problems. Some take advantage of special structures underlying the problems studied, while others make simplifying assumptions. There are specialized algorithms for well known problem classes such as linear programming, quadratic programming, network programming, discrete (or combinatorial) optimization, and multilevel optimization, just to name a few.

In the research project presented in this dissertation the goal is to develop an algorithm that can be applied to a wide class of nonlinear, constrained problems with only continuous variables or with both continuous and discrete variables. With no assumptions on convexity, these problems can (and often do) have multiple optima. Finding the best local optimum belongs to the realm of Global Optimization. By nature, Global Optimization problems are hard, especially if little is assumed about the problems in advance. To make the difficulties manageable, every approach of Global Optimization, be it deterministic,

stochastic, or heuristic, makes at least some assumptions about the problems it attempts to solve.

The OptQuest/GRG (OQGRG) algorithm was developed with the large target class of smooth mixed integer, nonlinear, constrained optimization problems in mind, meaning problems where all problem functions are differentiable in the continuous variables. The smoothness assumption is required in order to a gradient-based local NLP solver be applicable. The problems are assumed to have a finite number of local solutions. With the discrete variables fixed, the nonlinear, constrained sub-problem is smooth and once differentiable. The problem domain is closed and all variables have finite upper and lower bounds. This is required by the OptQuest Scatter Search algorithm.

OQGRG is a multi-start, 2-stage, meta-heuristic global optimization algorithm that combines the efficiency of Scatter Search in the global search phase and the power of a reduced gradient nonlinear solver in the local search phase. OQGRG uses OptQuest as the implementation of Scatter Search and Lsgrg2 as the implementation of a reduced gradient solver. It was written in standard ANSI C, using the Microsoft Visual C++ 6.0 development environment. A GAMS interface provides access to many test problems for which optimum or best known solutions are readily available in the literature.

In Chapter 2, the discussion first focuses on the background of Global Optimization with references to deterministic and stochastic approaches. The

meta-heuristic Scatter Search and its implementation, OptQuest, are described in detail in Chapter 3. Chapter 4 presents information about the properties of different constrained nonlinear local solvers, especially reduced gradient based methods, with Lsgrg2 as the implementation used in OQGRG. Details of the OQGRG algorithm itself, with options, parameters, and variations, are discussed in Chapter 5. Computational results for over 160 nonlinear problems, most of them continuous and constrained, some mixed integer, are shown in Chapter 6. Finally, Chapter 7 concludes the discussion with a summary and an outlook on opportunities for future research. Lists and tables are in the Appendix and the Bibliography has references to related papers and books from the literature.

2. Review of Global Optimization

Optimization problems from science, engineering, and management often cannot be described with only linear, continuous variables. These problems can (and often do) have multiple local optima. Global Optimization (GO) approaches aim to find the best overall solutions. Typically, the number of local solutions is unknown and can be quite large. Furthermore, the quality of the various local solutions may differ significantly. Global Optimization problems can be quite difficult. Most classes of global optimization problems are NP hard. This implies that it can be reasonably expected that for every problem type and algorithm there exist problem instances that take more than polynomial effort to solve in terms of the problem size.

Global optimization is one of the most challenging part of nonlinear programming. Even the most advanced mathematical programming and scientific computing environments lack a general, proven direct solver capability to handle continuous problems with many variables that have multiple optima. For promising efforts see the MLSL solver from Frontline Systems, Inc. (www.frontsys.com) and the new GLOBAL World site on GAMS World (www.gamsworld.org/global).

The form of the general problem (*P*) is as follows:

$$\text{Minimize } f(x) \tag{2.1}$$

$$\text{Subject to } x \in S, \tag{2.2}$$

where *f* is a continuous function on *S*, and $S \subset \mathbf{R}^d$ is a compact set with dimension *d*. Some methods may have further assumption on the objective function *f* or the feasible region *S*. These additional assumptions are rather weak and under these conditions an optimal solution exists and is attained. In other words, if

$$f^* \equiv \min_{x \in S} f(x)$$

then the set of optimal solutions

$$S^* \equiv \{ x \in S : f(x) = f^* \}$$

is nonempty.

It is well known that the general global optimization problem (P) is unsolvable in a finite number of steps. Therefore the global optimization problem is considered solved if a point is found in the neighborhood defined as

$$B_\varepsilon(S^*) \equiv \{x \in S : \| x - x^* \| \le \varepsilon \text{ for some } x^* \in S^*\}$$

or in the level set

$$S_\varepsilon \equiv \{x \in S : f(x) \ge f^* - \varepsilon\} \text{ for some } \varepsilon > 0.$$

There are several main classes of GO approaches that possess strong theoretical convergence properties. All such rigorous approaches have an inherent

computational demand that increases exponentially as a function of problem size. Global convergence can be guaranteed only if the global search component of the algorithm is used, at least in theory, in a complete, exhaustive fashion. Several GO classes an approaches are described below.

2.1 Exact Methods

'Naïve' Approaches

Sequential GO strategies, like the uniform grid search or the space covering sampling belong to this category. These methods are obviously convergent under mild assumptions, but they are 'hopeless' in higher dimensional problems. See details in [Horst and Pardalos, 1995] and [Pintér, 1996].

Complete Search Strategies

Mostly applicable to combinatorial and certain 'well-structured' problems (i.e. concave programming), these approaches are based on a complete evaluation of the possible solutions. For examples see [Horst and Tuy, 1993], [Horst and Pardalos, 1995] and [Pintér, 1996].

Branch and Bound Algorithms

Several different versions of partitioning strategies have been proposed to exactly solve GO problems. They are based on iteratively applying adaptive

partitioning, sampling, and subsequent lower and/or upper bounding procedures to the collection of active remaining subsets within the whole feasible set *S*. The exhaustive search feature is similar to the known integer programming methodology. Many approaches and a variety of implementations belong to this category.

These methods typically rely on structural knowledge of the problem, such as the maximum/minimum rate of each function (the objective *f* and constraints *g*), i.e. the overall Lipschitz constant, or the analytic formulation of all functions. The general Branch and Bound method is applicable to a variety of GO problems, such as concave minimization, DC programming, reverse convex programs, Lipschitz optimization, and combinatorial optimization. For further details see [Horst and Tuy, 1993], [Pintér, 1996] and [Pardalos, Romeijn and Tuy, 2000].

2.2 Stochastic Approaches

The following methods contain some stochastic elements, and as a consequence, either the outcome of a method itself is a random variable or the objective value is a realization of a stochastic process. With this, the possibility of an absolute guarantee of success is sacrificed. Instead, the aim now is to prove that as the effort increases to infinity, an element of $B_\varepsilon(S^*)$ or S_ε is found with probability one.

Pure Random Search (PRS)

The simplest algorithm for solving the global optimization problem consists of only one phase: generating a sequence of i.i.d. uniform points in the feasible region *S*, while keeping track of the best point found. A brief description of the algorithm is as follows.

Step 0. Set n=1, $y_0 = \infty$.

Step 1. Generate x from the uniform distribution over *S*.

Step 2. If $f(x) < y_{n-1}$, set $y_n = f(x)$ and $x_n = x$. Otherwise, $y_n = y_{n-1}$, and $x_n = x_{n-1}$.

Step 3. Increment n, return to Step 1.

The probability that after n iterations a point in the set $B_\varepsilon(S^*)$ is found is

$$\Pr(X_i \in B_\varepsilon(S^*) \text{ for some } 1 \le i \le n) = 1 - \Pr(X_i, \ldots X_n \notin B_\varepsilon(S^*))$$

$$= 1 - (1 - \mathbf{U}(B_\varepsilon(S^*)))^n,$$

where **U** denotes the uniform distribution on *S*. Since *f* is continuous, $\mathbf{U}(B_\varepsilon(S^*)) > 0$, and the above probability tends to 1 as $n \to \infty$. Thus the PRS algorithm offers a probabilistic asymptotic guarantee. It can also be proven that Y_n converges to the global optimum with probability 1 with increasing *n*, i.e.

$$\Pr(\lim_{n\to\infty} Y_n = f^*) = 1.$$

The PRS algorithm is very inefficient. In general, the expected number of points generated before one will fall within $B_\varepsilon(S^*)$ increases exponentially with

the dimension *d* of the problem. Extensions to this idea have been proposed that also use the generation of uniformly distributed points over *S* and some kind of local search from the points in the sample, or from a subset of the points in the sample.

For the extensions discussed below, some further assumption are made. There exist only a finite number of local optima for the problem, and they are all located in the interior of *S*. Additionally, it is usually assumed that *f* is twice continuously differentiable.

Two-phase methods

The methods discussed here have two phases. First, a *global phase*, in which the function is evaluated in a number of randomly sampled points. Second, a *local phase*, in which the sample points from the global phase serve as staring points to some kind of intensification method, i.e. local search. The points resulting from the local searches are local optima, and serve as candidates for the global optimum. Most of the methods are some sort of a variation on the original Multi-Start algorithm. In this algorithm, the global phase consists of generating uniformly distributed random points over *S*. Then a local search procedure *L* is applied to each of these points, resulting in a set of local optima.

Multi-Start

The simplest way of using a local search procedure is implemented in the original Multi-Start method.

Step 0. Set n=1, $y_0 = \infty$.

Step 1. Generate x from the uniform distribution over S and apply L to x, resulting in x'.

Step 2. If $f(x') < y_{n-1}$, set $y_n = f(x')$ and $x_n = x'$. Otherwise, $y_n = y_{n-1}$, and $x_n = x_{n-1}$.

Step 3. Increment n, return to Step 1.

Multi-Start is more efficient than pure random search in finding local optima, but a serious drawback is that each local optimum will inevitably be found several times. Assuming that the local search L is time consuming, it should ideally not be invoked in the *region of attraction* of any local optimum more than once. (The region of attraction of local optimum x_k^* is defined as the set of points in S starting from which L will converge to x_k^*.) The following algorithms have been designed with this objective in mind.

Clustering Methods

Clustering methods start from a uniform sample from S and create groups of these points that are considered "close" to each other by some measure. L is started no more than once in each of these groups, which will significantly reduce

the effort spent on finding local optima by the local search L. The most common way of creating the groups from the initial sample is by means of reduction, where only a fraction of the points with the best objective values are retained. The advantage of performing reduction is that the remaining points will still be uniformly distributed and are more "promising" as starting points for L. Also, points in the clusters will belong to the same level set. As a consequence, clustering rules can be derived that will result in methods with good statistical properties regarding convergence to the global optimum.

Density Clustering

In this method, points from the reduced sample are assigned to clusters based on their distance from the local optima found so far. The critical distance function, upon which the decision of assigning a point to a cluster is made, is dependent on several things: on the total number of points generated, on the dimension of the problem, and on the Hessian at the point. The idea behind including the latter in the critical distance calculation originates from the assumption that function f is locally approximated by a quadratic function, i.e. the neighborhood of a local optimum can be approximated by an ellipsoid. For more details on the formula for r_k and its derivation see [Rinnooy Kan and Timmer, 1987a].

Single Linkage Clustering

The Single Linkage method does not rely on an assumption made on the shape of the cluster like the Density Clustering method does. It forms clusters from the reduced sample points x sequentially based on the distance

$$d(x, C) = \min_{y \in C} \|x - y\|.$$

The point x with minimal distance from the nearest point already belonging to cluster C is added to the cluster. (The seed for the first cluster is a result of L, started on the point with the best value from the first reduced sample.) This process is performed until the distance d is larger than a critical value r_k and no more points can be added to any cluster. If there is a point that has not been added to any cluster, L is started from it. It either finds a new local solution, which will serve as a seed in subsequent clustering, or if it finds an already known seed, the point is added to that cluster. When all points from the reduced sample are assigned to a cluster, new points are generated and the clustering process starts again with an updated r_k. For more details on the Single Linkage Clustering, see [Rinnooy Kan and Timmer, 1987a].

Experience suggests that Single Linkage Clustering approximates the level sets better than Density Clustering. By appropriately selecting a constant in the formula to calculate r_k, the probability of starting the local search L tends to zero as the number of iterations of generating new random samples goes to infinity. Furthermore, the probability that the local search L is started in a finite number of

times will converge to one, even if the sampling continues forever. This very attractive property is overshadowed by the loss of the asymptotic guarantee of success. In the Single Linkage Clustering it is possible to create a cluster with the level set $\{ x \in S : f(x) \geq y_\gamma \}$ that contains more than one region of attraction. It is possible to miss some local optima and thus the global optimum.

Multi Level Single Linkage

The Multi Level Single Linkage (MLSL) method developed by [Rinnooy Kan and Timmer, 1987b] combines the theoretical advantages of the Multi-Start method and the computational efficiencies of the clustering methods. Here the local search L is applied to every sample point except if there is another sample point with better function value within a critical distance. The following is a brief description of the algorithm from [Boender and Romeijn, 1995]:

Step 0. Set k=1, X^* = {empty set}.

Step 1. Generate N points, $x_{(k-1)N+1}, \ldots x_{kN}$, from the uniform distribution over S. Set i=1.

Step 2. If $\exists j$ s.t. $f(x_j) < f(x_i)$ and $\|x_j - x_i\| < r_k$ then go to Step 3. Otherwise, apply L to x_i , and add the local optimum found to X^*.

Step 3. Increment i. If $i \leq kn$, go to Step 2. Otherwise, increment k, and go to Step 1.

The critical distance chosen by [Rinnooy Kan and Timmer, 1987b] is

$$r_k = \frac{1}{\sqrt{\pi}}(\Gamma(1+\frac{d}{2}) * m(S) * \frac{\zeta \ln(kN)}{kN})^{\frac{1}{d}} .$$

The algorithm has some very strong theoretical properties:

- For any arbitrary starting point, the probability that L is started from it tends to 0 as $k \to \infty$.
- If ζ >2, the probability that the local search L is applied in iteration k tends to 0 as $k \to \infty$.
- If ζ >4, the total number of local searches L started by MSLS is finite with probability 1.
- Any local optimum will be found by MSLS within a finite number of iterations with probability 1.

Since the critical distance decreases with increasing iteration number k, a point from which L was previously not started may become a starting point in the next cycle. Hence, all sample points generated must be saved, which increases the amount of storage required and complicates the implementation of the algorithm. It also makes the choice of the sample size, N, important, since too small a sample leads to many revised decisions, while too large a sample will cause L to be started many times.

Random Linkage (RL)

Recently [Locatelli and Schoen, 1999] introduced a class of "Random Linkage" (RL) multi-start algorithms that retain the good convergence properties of MLSL and do not require that past starting decisions be revised. Uniformly distributed points are generated one at a time, and L is started from each point with a probability given by a non-decreasing function $\phi(l)$, where l is the distance from the current sample point to the closest of the previous sample points with a better function value. Assumptions about this function that give RL methods the same theoretical properties as MLSL are derived in the above reference.

An Implementation of MLSL for Constrained Problems

Fylstra et. al. have recently implemented a version of MLSL which can solve constrained problems of the following form. For more information on the extended solver platform from Frontline Systems, Inc. visit [www.frontsys.com].

$$\text{Minimize} \quad f(x) \tag{2.3}$$

$$\text{Subject to} \quad gl_i \le g_i(x) \le gu_i \qquad i=1..m \tag{2.4}$$

$$x \varepsilon S \tag{2.5}$$

It uses the L_1 exact penalty function, defined as

$$P_1(x,w) = f(x) + \sum_{i=1}^{m} \left(w_i * viol(g_i(x)) \right) \qquad (2.6)$$

where w_i's are nonnegative penalty weights. The function $viol(g_i(x))$ is equal to the absolute amount by which the *i*th constraint is violated at the point *x*. It is well known (see [Nash and Sofer, 1996]) that if x^* is a local optimum of (2.3)-(2.5), u^* is a corresponding optimal multiplier vector, and the second order sufficiency conditions are satisfied at (x^*, u^*), then if

$$w_i > abs(u_i^*)$$

x^* is a local unconstrained optimum of P_1. If (2.3)-(2.5) has several local optima, and each w_i is larger than the maximum of all absolute multipliers for constraint *i* over all these optima, then P_1 has a local optimum at each of these local constrained optima. Even though P_1 is not a differentiable function of *x*, MLSL can be applied to it, and when a randomly generated trial point satisfies the MLSL criterion to be a starting point, any local solver for the smooth NLP problem can be started from that point. The local solver need not make any reference to the exact penalty function P_1, whose only role is to provide function values to MLSL. Without any theoretical investigations of this extended MLSL procedure, it must currently be regarded as a heuristic.

Other Stochastic Methods

All of the methods discussed so far in this chapter aim to find all local optima. There are other methods that directly aim at finding the global optimum without finding all the local optima. The best-known methods belonging to this group are variants of the Adaptive Search and different versions of Simulated Annealing. It has been shown that Simulated Annealing is basically an approximation of the Adaptive Search [Boender and Romeijn, 1995].

Another alternative approach for solving the global optimization problem is the *random function* approach. This approach is most useful for optimization of objective functions *f* that are very expensive to evaluate. The function *f* is considered to be a realization of an a-priori defined stochastic process, so the properties of *f* depend on the properties of the stochastic process. Difficulty in this approach arises from the requirements that while the stochastic process should be defined to be as consistent as possible with the known properties of *f*, the process should be mathematically tractable as well. It turns out that in most cases, the tractable a-priori stochastic processes have non-differentiable, one-dimensional sample paths, whereas for stochastic processes that approximate *f* at an acceptable level, the a-posterior distribution obtained from the processes are hardly ever explicit, manageable expressions. An excellent brief overview of the above mentioned methods can be found in [Boender and Romeijn, 1995].

3. Scatter Search and the OptQuest Callable Library

Scatter Search (SS) is a population based meta-heuristic algorithm devised to intelligently perform a search on the problem domain [Glover, 1998]. It operates on a set of solutions called the *reference set* or *population*. Elements of the population are maintained and updated from iteration to iteration.

Philosophically, Scatter Search differs from other population based evolutionary heuristics like Genetic Algorithms (GAs) mainly in its emphasis on generating new elements of the population mostly by deterministic combinations of previous members of the population as opposed to the more extensive use of randomization. SS was founded on strategies that were proposed as augmentations to GAs more than a decade after their debut in Scatter Search. It embodies principles and strategies that are still not emulated by other evolutionary methods and prove to be advantageous for solving a variety of complex optimization problems.

SS operates on a set of points, called reference points, which constitute good solutions from previous solution efforts. The approach systematically generates new combinations of the reference points to create new points, each of which may be mapped into an associated feasible point. The process of generating new points balances two important, but often contradictory requirements: intensification and diversification of the new points. Intensification is a natural

goal in obtaining better solutions in the neighborhood of solutions with already good objective value. Diversification is desirable in helping to avoid myopically concentrating on locally good solutions without searching for good solutions in distant, yet unexplored regions of the problem domain.

The following are the steps of SS, as implemented in OptQuest [Laguna and Marti, 2001] and [Laguna and Marti, 2000]. The problem to be solved, in OptQuest's terminology, is: minimize *f(x)* subject to linear constraints, non-linear requirements, and simple bounds on the variables. The bounds define a rectangle *S*.

3.1 Steps of Scatter Search

1. Initialize: size of reference set = *b*, initial point =*x0*, input upper and lower bounds on variables and constraint functions, and the coefficients of any linear constraints. Create an initial set of three points, $R0 \subset S$, the first with all variables set to the lower bound, the second with all variables set to the upper bound, and the third with all variables set to the mid-point between the bounds. If an initial point has been determined, add it to *R0*.
2. Given *R0,* use a *diversification generation method* to augment it with additional points, creating an initial diverse reference set, $R \subset S$ of cardinality *b*. Optionally, map the elements of *R* into points that satisfy the linear constraints.

3. Evaluate the objective *f* and the nonlinear constraint functions *G* at each point in *R*, and use these values to form a penalty function P_{OQ}, which is formed by multiplying the maximum percentage violation of the nonlinear constraints by a positive penalty weight and adding the resulting penalty term to the objective. This function is used as a merit function to order the population points by their quality.

While (stopping criteria are not satisfied)

While (some distinct pair of points in *R* has not been processed)

4. Select a new pair of points in *R*
5. Use a s*olution combination method* to produce a small number of trial solutions from this pair of points. Optionally, map each trial point into the closest point that satisfies the linear constraints and variable bounds.
6. At each (mapped) trial solution, evaluate the objective *f* and nonlinear constraint functions *G*, and form the penalty function, P_{OQ} .

Endwhile

7. Update the reference set.
8. If the reference set has changed, return to step 4. Otherwise, restart the procedure by selecting a subset (typically the best half) of the best points in the reference set to be retained as the set *R0*, and return to step 2.

Endwhile

3.2 Description of the OptQuest steps

Step 1 generates the starting points to create the initial reference set *R0*. The 3 points always appearing in this set are the vectors *x* for which all element are set to the upper bounds, to the lower bound, and to the midpoints of the bounds. If there is an initial point recommended to the problem, it is also added to *R0* as a fourth point.

Step 2 generates the remaining points to the initial reference set *R*. The *diversification generation method* begins by generating *nr>b* randomly generated points in *S*, using a stratified sampling procedure described in [Laguna and Marti, 2000]. It then creates the reference set, *R*, by adding to *R0* the random point farthest from its nearest neighbor in *R0*, and repeating this process until *R0* has cardinality *b*. If the problem has linear constraints and the points selected are infeasible for these linear constraints, they are first projected onto the convex polyhedron defined by the linear constraints and then added to *R0*. This is done by finding the point in this polyhedron which is closest (using the L1 norm) to the infeasible point by solving a linear program. The result of this step is a reference set that contains a diverse set of points that satisfies the linear constraints of the problem.

The initial population resulting from this procedure for a reference set of size $b = 10$ is shown in Figure 3.1, which uses data from a 2 variable unconstrained problem due to [Dixon and Szego, 1975] called the six-hump camelback function. The objective to be minimized is

$$F(x, y) = 4x^2 - 2.1x^4 + \frac{1}{3}x^6 + xy - 4y^2 + 4y^4 \qquad (3.1)$$

This is the problem EX8_2_5 from a large set of problems described in [Floudas, et. al., 1999]. Problems from this set are used as test problems for OQGRG, and will be discussed in detail later. The problem has upper bounds of 10.0 and lower bounds of –10.0 on both variables, and has 6 local minima, all lying well within these bounds (see Figure 3.2 for their location), plus a stationary point at the origin that is neither a local minimum nor a maximum. The initial set *R0* is the three points (0,0), (10,10), (-10,-10), where (0,0) is user-supplied and the other two are the vectors of upper and lower bounds respectively.

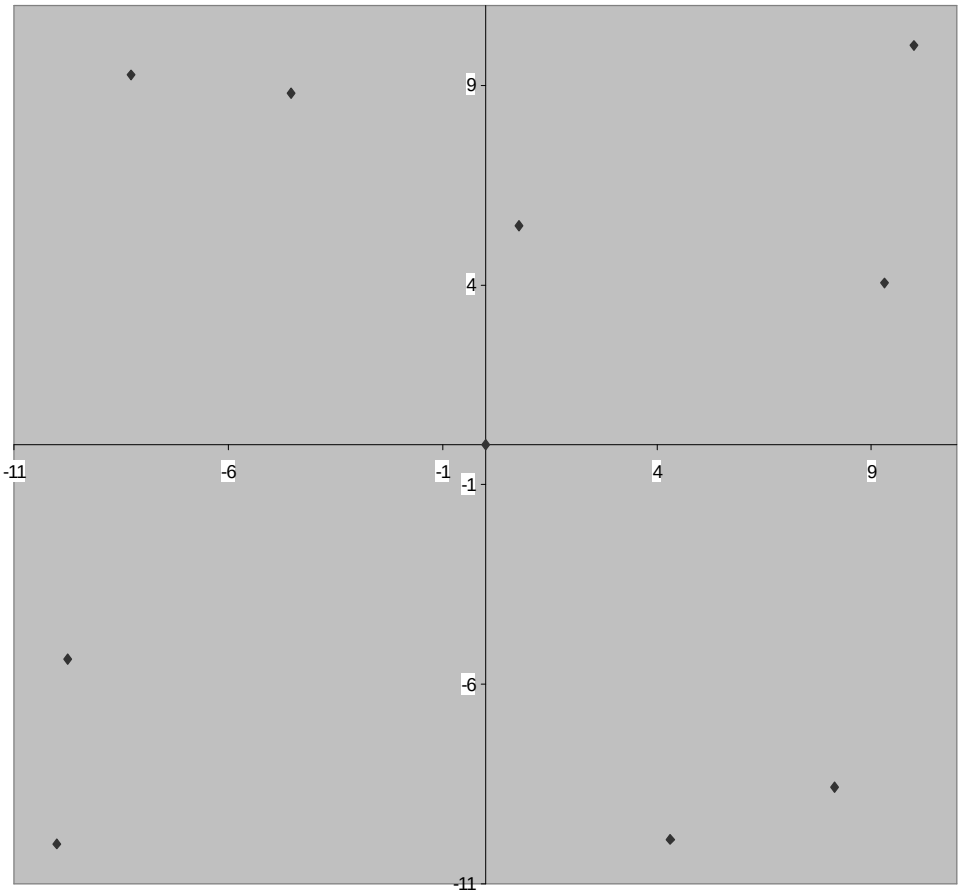


Figure 3.1: Initial Population for the Six-Hump Camelback Function

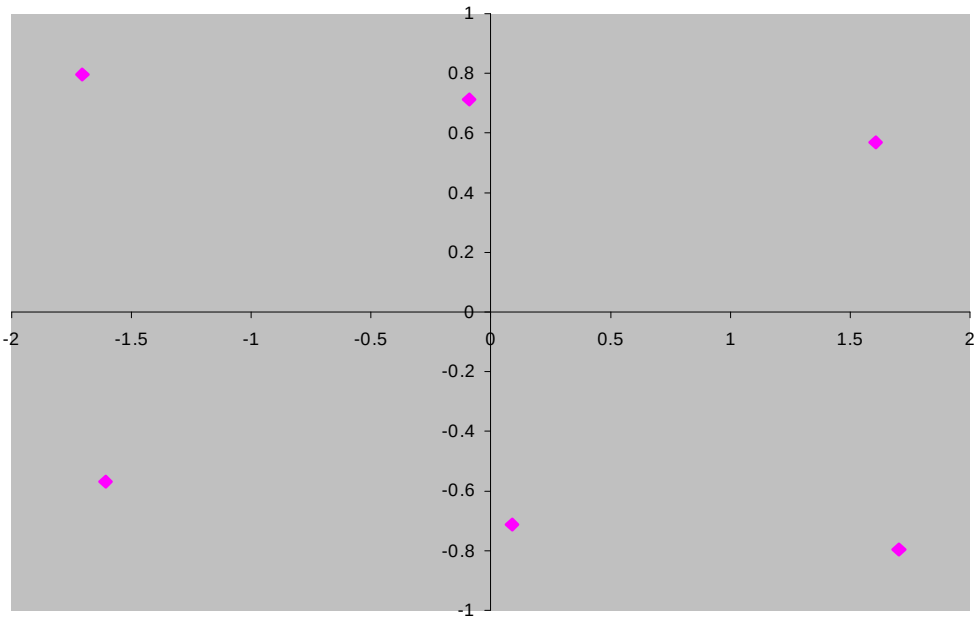


Figure 3.2: Locally Optimal Solutions for the Six-Hump Camelback Function

Step 3 ranks the points in the reference set based on their quality, measured by a penalty function P_{OQ} which is equal to the objective plus a penalty weight times the maximum percentage violation of the violated nonlinear constraints. The penalty function P_{OQ} is not the same as the exact penalty function P_1 described in (2.6), and is not exact. It is used because Lagrange multiplier information is assumed not to be available. Multipliers may not even exist if the problem is non-smooth or has discrete variables, and OptQuest is designed to solve such problems as well.

Steps 4 and 5 create new trial solutions by selecting 2 “parent” points from the reference set and performing the solution combination method on them. (In case the resulting points do not satisfy the linear constraints of the problem, their projection onto the convex polyhedron will take their place.) To illustrate how the combination method currently implemented in OptQuest works, Figure 3.3 demonstrates the generation of new trial points from the 3 best points of the initial population for the six-hump camel back function.

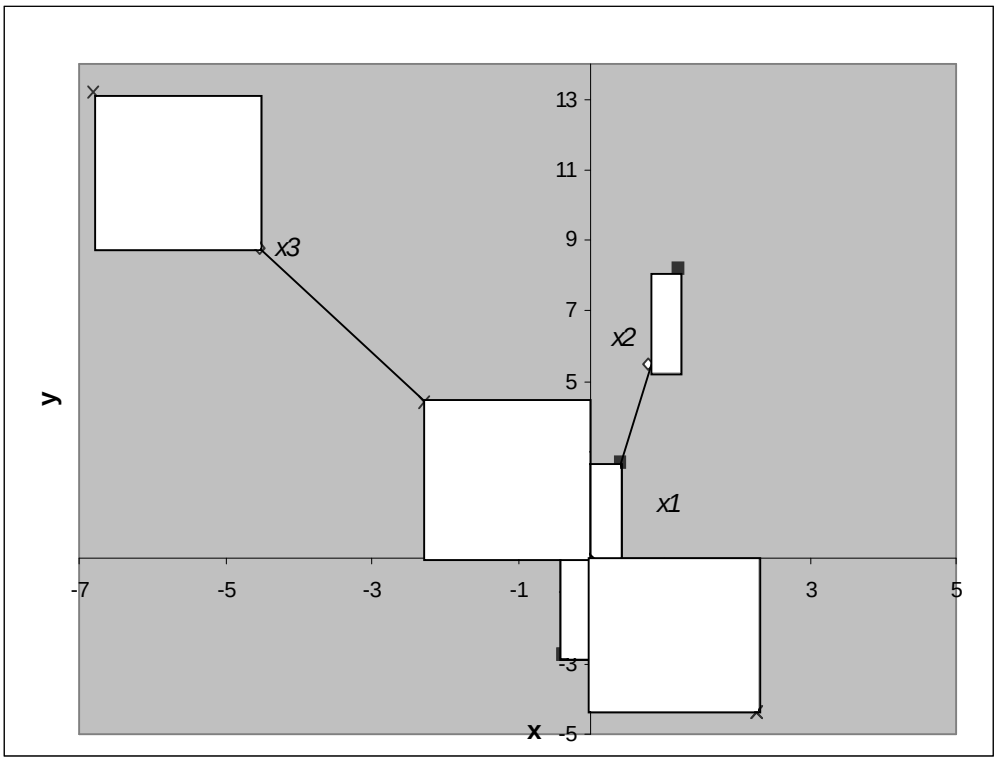


Figure 3.3: Trial Points for Six-Hump Camelback Function

The three points shown as diamonds and labeled *x1, x2, x3* (*x1* is the origin) have the lowest objective values in the initial population. The two lines in the figure are determined by the pairs of points *(x1, x2)* and *(x1, x3)*. Focusing on the line *(x1, x2),* let

$$
\begin{aligned}
d &= (x2-x1)/2 \\
v1 &= x1-d \\
v2 &= x1 \\
v3 &= x1+d \\
v4 &= x2 \\
v5 &= x2+d
\end{aligned}
\tag{3.2}
$$

Thus *v3* is at the midpoint of this line, and *v1* and *v5* extend it beyond *x1* and *x2*. These points are shown as "x" in the figure. The points *v1* and *v2* can be used to define a hyper-rectangle whose 2^n vertices are the set

$$V = \{(z_1, z_2, ..., z_n) \mid z_i = (v1)_i \, or \, (v2)_i, i = 1, ..., n\} \tag{3.3}$$

Thus the four pairs of points (*vi, v(i+1))* for $i = 1,..,4$ define 4 rectangles, three of which are shown on the line determined by *x1* and *x2*. OptQuest generates one randomly distributed trial point in each rectangle, and these points are shown as triangles. Three more trial points are generated in the same way starting with the points (*x1, x3)*. These points lie "close" to the lines, but are not on them.

If there are discrete variables, the above process produces trial points whose values for these variables do not lie in the finite set of allowed values.

These components are rounded to an allowable value using generalized rounding processes, i.e. processes where the rounding of each successive variable depends on the outcomes of previous roundings.

The full set of 144 trial points generated from the initial population of this example is shown in Figure 3.4.

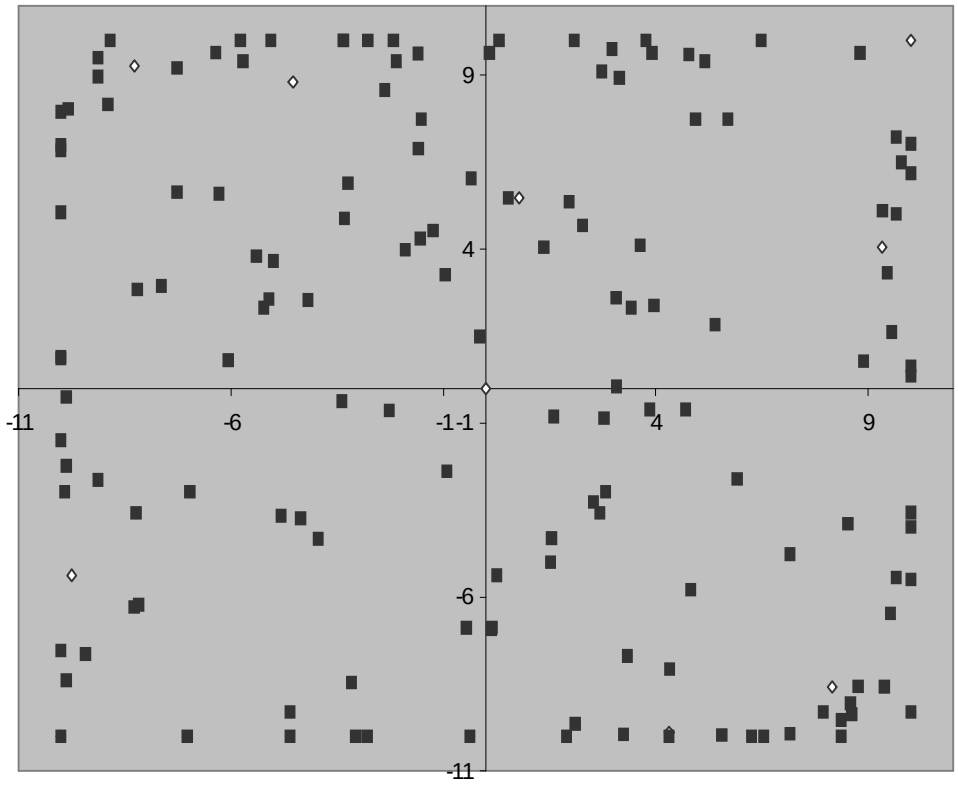


Figure 3.4: Initial Population and 144 Trial Points for Six-Hump Camelback Problem

The ten white diamond points are the members of the initial reference set, while the dark squares are the trial points, generated as described earlier. These are well scattered over the region defined by the bounds.

In Step 6 the objective f and non-linear constraints G are evaluated and the penalty function P_{OQ} is calculated. OptQuest considers f and G to be black boxes, and it is the responsibility of the user to provide the evaluation and return the corresponding values.

In step 7, after all trial points have been evaluated, the reference set is updated by replacing the population which generated the trial points by the best b points of the union of the trial points and the initial reference set, where best is determined by the OptQuest penalty function P_{OQ}. This is an aggressive update, emphasizing solution quality over diversity. This updated reference set, used to generate trial points from iteration 155 onward, is shown in Figure 3.5. The ten population points cluster in the region about the origin where the six local optima are located, so the next set of 144 trial points will lie within a slight expansion of this region. These trial points thus have much better objective values than those generated by the initial population, as we illustrate later in section 6.

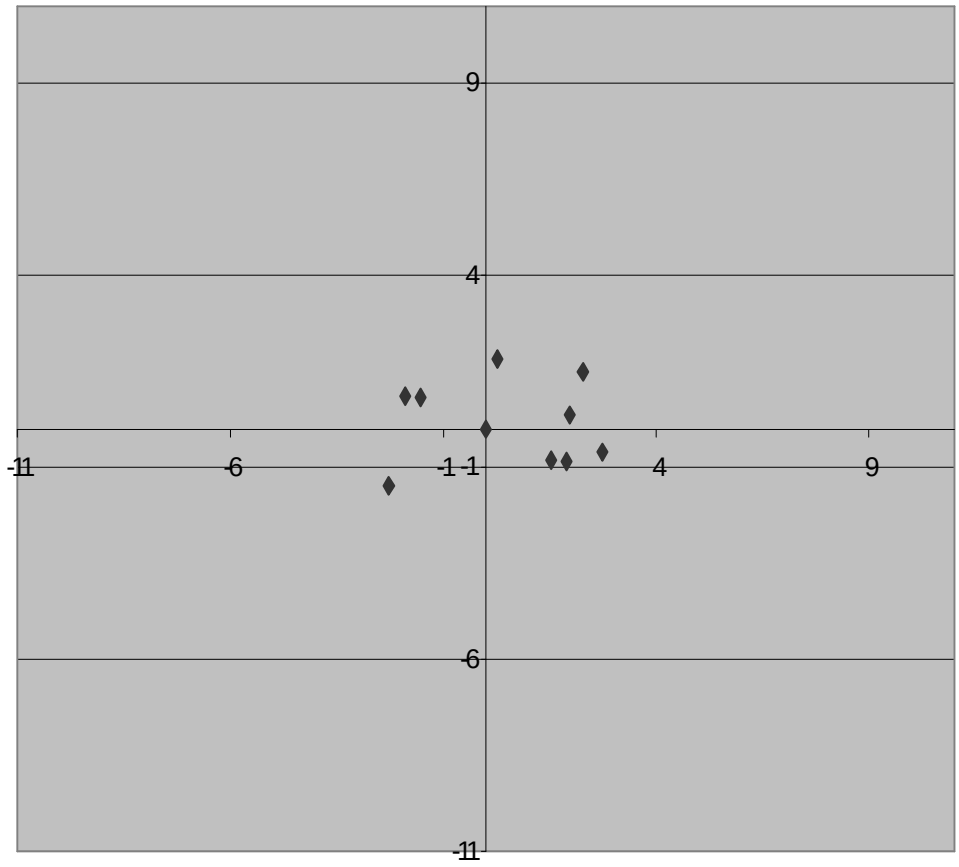


Figure 3.5: Second Reference Set for Six-Hump Camelback Function

When the diversity of the reference set is considered equally important to the quality of the solutions, a different updating method is suggested in [Laguna and Marti, 2001]. In this variation, the reference set is split into 2 halves. The first half is created and maintained the same way as described earlier, focusing on the quality of the points. The other half contains diverse points. If a solution does not

qualify to enter the first half of the reference set based on its quality, a test is performed to determine whether it fits the diversity criterion. That is, if the new point's minimum distance to any point in the second half of the reference set is larger than that of any points' already in the set, the new point will replace it. With this method the dynamic preservation of diversity is assured for the reference set.

It is possible that after a number of cycles of generating new trial solutions there is no improvement in the best solution found. In this case, step 8 forces a return to Step 4 where a new diverse reference set is created. When this occurs, a number (typically $b/2$) of the best points from the current population are retained and newly generated points replace the rest.

Precision of the Objective and the Solutions

Improvements in the solutions are observed if the absolute differences between the objective values of two solutions exceed the precision set for the objective. The default value of the objective precision is 10^{-4}. Similarly, two solutions are considered different if the absolute difference between at least one of their decision variables is greater than the variable precision. The default value of the variable precision is also 10^{-4}.

During the search process OptQuest can check if newly generated trial solutions have already been proposed. The purpose of this feature is to avoid

evaluating trial solutions more than once. OptQuest maintains a database of the evaluated trail solutions and performs a check to find if a new solution is a duplicate. The variable precision value is used to compare the decision variables of solutions.

By default, checking for duplicates in the database is enabled, with the default size of the database set for 10,000. The database-check option can be turned off or the user can change the size of the database.

3.3 OptQuest: the Implementation of Scatter Search

OptQuest is available as a static callable library written in C, which can be invoked from any C program, or as a dynamic linked library (DLL) which can be called from a variety of languages including C, Visual Basic, and Java. The callable library consists of a set of functions which (a) input the problem size and data, (b) set options and tolerances, (c) perform steps 1 through 3 to create an initial reference set, (d) retrieve a trial solution to be evaluated in step 6 above and, (e) communicate these objective and constraint values back to OptQuest, which uses them as the input to step 7 above. For a complete description, see [Laguna and Marti, 2001].

4. Local Optimizers for NLP Problems

A key element in the OptQuest/GRG approach of solving Mixed Integer Non-linear Programming (MINLP) problems is the use of a local optimizer *L* that can find a local solution to the non-linear optimization problem under fairly general smoothness and regularity conditions.

The general form of the non-linear problems we need to solve in the OptQuest/GRG scheme is the following:

$$\text{Minimize} \quad f(x) \tag{4.1}$$

Subject to the (nonlinear) constraints

$$gl \le G(x) \le gu \tag{4.2}$$

$$x \varepsilon S \tag{4.3}$$

where x is an n-dimensional vector of continuous decision variables, $G(x)$ is an m-row vector of linear and nonlinear constraints, and the vectors gl and gu contain upper and lower bounds for the (possibly nonlinear) constraints. The set S is defined by simple bounds on x, and it is assumed to be closed and bounded. This property is a result of the OptQuest scatter search algorithm's requirements on the variable domain. The objective function f and the m-dimensional vector of constraint functions G are assumed to have continuous first partial derivatives at all points in S.

The main classes of algorithms in widespread use today are Successive Quadratic Programming (SQP) and Generalized Reduced Gradient (GRG) – see [Edgar, Himmelblau, and Lasdon, 2001], Chapter 8. The algorithm implemented in the widely used MINOS solver [Murtagh and Saunders, 1982] is similar to SQP. Random search methods have a long history in nonlinear optimization [Ugray, 1979], and some of the adaptive search techniques have lately gained importance in Global Optimization as well – see more in Chapter 2.

Gradient-based NLP solvers rapidly converge to the "nearest" local solution, and can easily achieve 4 to 8-digit accuracy. They can also efficiently handle equality constraints (more generally, narrow feasible regions). An additional advantage of Generalized Reduced Gradient (GRG) methods from the perspective of being a component in a multi-start global optimizer is their 2-phase structure: they attempt to become feasible first before finding the optimum. This feature causes them to be more reliable in finding a feasible solution and diagnosing infeasibility than other methods (for example SQP and MINOS), which attempt to achieve both feasibility and optimality in a single phase.

If there are nonlinear constraints, SQP and MINOS generate a sequence of points that usually violate the nonlinear constraints, with the violations decreasing to within a specified feasibility tolerance as the sequence converges to a local optimum. GRG algorithms have a simplex-like phase 1- phase 2 structure. Phase 1 begins with the given starting point and, if it is not feasible, attempts to find a

feasible point by minimizing the sum of constraint violations. If this effort terminates with some constraints violated, the problem is assumed to be infeasible. However, this local optimum of the phase 1 objective may not be global, so a feasible point may exist. If a feasible point is found, phase 2 uses it as its starting point, and proceeds to minimize the true objective. Both phases consist of a sequence of line-searches, each of which produces a feasible point with an objective value no worse (and usually better) than its predecessor.

There are several parameters and options that strongly influence the reliability and efficiency of a GRG implementation. The *feasibility tolerance* (*ft*; default value 10^{-4}) determines when a constraint is satisfied. If the constraint has the form $g(x) \geq l$, it is considered satisfied in Lsgrg2 (the GRG code used in the Optquest/GRG implementation) if

$$abs(g(x) - l) \geq -ft(1.0 + abs(l)). \tag{4.4}$$

The optimality tolerance (*ot*; default value 10^{-4}) and a number of consecutive iterations (*nstop*; default value 10) determine when the current point is declared optimal. This occurs when

$$kterrnorm \leq ot \tag{4.5}$$

where *kterrnorm* is the infinity norm of the error in the Kuhn-Tucker conditions, or when

$$abs(f(x_k) - f(x_{k+1})) \leq ot(1.0 + abs(f(x_k)) \tag{4.6}$$

for *nstop* consecutive values of the iteration index, *k*.

There is also a scaling option, which determines row and column scale factors so that the absolute values of the nonzero elements of the scaled Jacobian matrix (evaluated at the starting point) are as close to unity as possible. For some problems, it may be necessary to use this scaling option to obtain the optimum solution.

There are many papers and texts discussing gradient based NLP solvers, e.g. [Nash and Sofer, 1996], [Nocedal and Wright, 1999], and see [Edgar, Himmelblau, and Lasdon, 2001]. Several good commercially available implementations of GRG and SQP solvers exist - see [Nash, 1998] for a review. As with any numerical analysis software, a local NLP solver can fail to find a local solution from a specified starting point. The problem may be too badly conditioned, badly scaled, or too large for the solver, causing it to terminate at a point (feasible or infeasible), which is not locally optimal. While the reliability of the best current NLP solvers is quite high, these difficulties occurred several times in computational testing, which will be discussed in more detail with the test results later.

Let L be a local NLP solver capable of solving (4.1)-(4.3), and assume that L converges to a local optimum for any starting point $x_0 \in S$. Let $L(x_0)$ be the locally optimal solution found by L starting from x_0, and let x_i^*, $i = 1,2,...,nloc$ be all the local optima of the problem. The basin of attraction of the ith local

optimum relative to L, denoted by $B(x_i^*)$, is the set of all starting points in S from which the sequence of points generated by L converges to x_i^*.

Formally:

$$B(x_i^*) = \{x_0 \mid x_0 \in S, L(x_0) = x_i^*\}$$

One measure of difficulty of a global optimization problem with unique global solution x_1^* is the volume of $B(x_1^*)$ divided by the volume of the rectangle, S, the *relative volume* of $B(x_1^*)$. The problem is trivial if this relative volume is 1, when the problem has one local optimum which is the global optimum as well. The best known class of problems for which this is known to be true are convex programs. The problem becomes increasingly difficult as the relative volume of $B(x_1^*)$ decreases. NLP solvers might be efficient in finding the local optimum once started inside the attraction basin, but the difficult task of selecting a starting point for them is the task of the global search phase.

5. The OptQuest/GRG (OQGRG) Algorithm

5.1 The Problem Statement

The most general problem this algorithm can solve has the form

Minimize $f(x,y)$ (5.1)

Subject to the nonlinear constraints

$$gl \le G(x, y) \le gu \tag{5.2}$$

and the linear constraints

$$l \le A_1 x + A_2 y \le u \tag{5.3}$$

$$x \varepsilon S, y \varepsilon Y \tag{5.4}$$

where x is an n-dimensional vector of continuous decision variables, y is a p-dimensional vector of discrete decision variables, and the vectors gl, gu, l, and u contain upper and lower bounds for the nonlinear and linear constraints respectively. The matrices A_1 and A_2 are m_2 by n and m_2 by p respectively, and contain the coefficients of any linear constraints. The set S is defined by simple bounds on x, and it is assumed to be closed and bounded, i.e. each component of x has a finite upper and lower bound. This is required by the OptQuest procedure. The set Y is assumed to be finite, and is often the set of all p-dimensional binary or integer vectors y. The objective function f and the m_1-dimensional vector of constraint functions G are assumed to have continuous first partial derivatives at

all points in $S \times Y$. This is necessary so that a gradient-based local NLP solver could be applied to the relaxed NLP sub-problems formed from (5.1) - (5.3) by allowing the *y* variables to be continuous.

5.2 OQGRG: A Combination of a Global Heuristic and a Local Search

Optquest/GRG is a two-phase, multi-start global optimization algorithm where the global phase employs the Scatter Search meta-heuristic and the local phase relies on a gradient-based non-linear optimizer. From a meta-heuristic point of view, one can look at the global phase as a method of ensuring diversification in the search process. In this view, the local phase is the equivalent of an intensification method.

Scatter Search generates *(x,y)* trial solutions that are within the problem domain (5.4) and satisfy the linear constraints (5.3). It attempts to find points that will minimize the objective (5.1), while trying to satisfy the non-linear constraints (5.2). Since Scatter Search does not include methods for generating points that satisfy the non-linear constraints, trial solutions for problems that have non-linear constraints with narrow feasible regions (non-linear equality constraints, for example) will often be infeasible. Although this may seem like a strong drawback of the method, it is not a big concern when we use these trial solutions only as

starting points for a local search phase where the local search procedure can efficiently move to feasible solutions, as gradient-based local optimizers do.

The non-linear optimizer takes a trial solution *(x,y)* from Scatter Search as a starting point, and attempts to find a locally optimal solution. Lsgrg2, the gradient-based NLP solver used in OQGRG is very efficient in finding locally optimal solutions that will satisfy the non-linear constraints (5.2) with high accuracy. The relative cost (in computational effort) of intensification of a good trial point, i.e. finding a local optimum through the use of the local solver, is high compared to the generation of trial solutions. Because of this, it is desirable to start the local solver as rarely as possible.

To reduce the effort spent in the local phase of the algorithm, trial solutions are filtered based on two criteria before a decision is made whether a trial solution is a good candidate as a starting point for the solver to find a new, better local solution. The Merit Filter will select trial points based on their quality, which is defined by the penalized objective value (the merit function value). Only high quality points, points with better merit function value than a dynamically changing threshold, pass this filter. The Distance Filter chooses from trial points based on their distances from already found local optima. A trial point can become a starting point for the local solver only if its distance from any previously found local solution is greater than a certain pre-determined fraction of the largest distance traveled to that local solution in previous local searches.

From a meta-heuristic point of view, the two filters can be considered as Tabu lists on the merit function value and the distance of trial points from already found solutions. Following the Tabu analogy, the criteria for the lists can change dynamically: threshold and distance values can be strengthened, as new solutions are found, or they can be relaxed if no new steps can be made for a substantial number of trials.

Trial solutions that pass both the Merit and Distance Filters become starting points of the local solver. Every new local optimum found is stored and algorithm parameters (i.e. merit threshold value, distance threshold value, merit function penalty weights) are dynamically updated as the search progresses.

5.3 Pseudo-Code of the Algorithm

Table 5.1 below describes the pseudo-code of OQGRG.

INITIALIZATION

1. Read_Problem_Parameters (n, p, m_1, m_2, bounds, starting point);
2. Setup_Algorithm_Parameters;
3. Setup_OptQuest_Parameters;
4. Initialize_OptQuest_Population;

STAGE 1: INITIAL OPTQUEST ITERATIONS AND FIRST GRG CALL

WHILE (unevaluated trial points generated from initial iteration limit remain)

{ 5. Get (trial solution from OptQuest);

6. Evaluate (objective and nonlinear constraint values at trial solution);

7. Put (trial solution, objective and constraint values to OptQuest database);

} (END_WHILE)

8. Get_Best_Trial_Point_from_Stage1 (starting point);

9. Call_GRG (starting point, local solution);

10. Calculate_Threshold (threshold, starting point);

STAGE 2: MAIN ITERATIVE LOOP

WHILE (stopping criteria not met)

{ 11. Get (trial solution from OptQuest);

12. Evaluate (objective and nonlinear constraint values at trial solution,);

13. Put (trial solution, objective and constraint values to OptQuest database);

14. Calculate_ Penalty_ Function (P_1, trial solution);

IF (Distance and Merit filter criteria are satisfied at trial solution)

{ 15. **Call_GRG (trial solution, local solution)**;

16. Analyze_Solution (GRG Terminating Condition);

17. Update_Local_Solutions_Found;

18. Update_Largest_Lagrange_Multipliers_Found;

19. Update_Merit_Filter_Parameters (*threshold*);

20. Update_Distance_Filter_Parameters (*maxdist*);

}

```
    ELSE IF (P1 > threshold for waitcycle consecutive
             iterations)
    {
          21. increase threshold;
    }
} (END_WHILE)
22. Report (solutions, statistics).
```

Table 5.1: The OQGRG Pseudo-code

5.3.1 Initialization

The first part of the algorithm reads in the problem parameters: continuous and discrete variables, linear and nonlinear rows, bounds, the optional starting point, and a few additional parameters, such as the indices of the objective row, objective value variable, and the sense of optimization. This is also the place to set up the algorithm and the OptQuest parameters. (Many parameters and algorithm options are controlled through a settings file; setting the parameters to other values than the defaults takes place here.) The algorithm parameters include the iteration limits Stage 1 and the entire algorithm, feasibility and optimality tolerances for GRG, threshold values, etc. OptQuest parameters include population size, iteration number, accuracy, list of continuous and discrete variables, bounds, linear and nonlinear constraints. The current settings file, with default values for all parameters, is shown in Appendix A.

Based on the problem, algorithm, and OptQuest parameters the initial population is generated before the iteration loops start.

5.3.2 Stage 1: Initial OptQuest Iterations and First GRG Call

The idea behind Stage 1 is to let OptQuest develop some 'knowledge' of the problem by exploring as much of the problem domain as possible. It is achieved by OptQuest recommending trial solutions for which the evaluation of the objective and nonlinear constraint values are returned to OptQuest's database.

At this early stage the trial solutions are combinations of points from the initial population, which is basically a diverse, random sample. The order in which parents for the trial solutions are selected from the population depends on the quality of the parent points. In this first cycle, however, this order is irrelevant from our perspective: we want OptQuest to explore the whole problem domain to search for high quality points.

At the end of Stage 1 the local solver, GRG, is called, starting from the best point found in Stage 1. Attributes of this best point will serve as a basis for many decisions made in the selection process of Stage 2. The merit function value (i.e. the sum of the objective and the L1 exact penalty term, which is the weighted sum of the constraint violations) of the best trial point is set as the threshold value for the Merit Filter. Also, for the local optimum found by GRG, the distance

threshold is set as the distance between the local optimum and the trial point from which GRG started.

5.3.3 Stage 2: Main Iterative Loop

The algorithm enters this stage after it has developed some 'knowledge' about the problem: by now OptQuest has performed an initial exploration of points widely scattered around in the problem domain and updated the reference set by the best points found. The local solver has been called from a promising starting point, the best trial point found so far. A *threshold* for the Merit Filter and a local solution with its associated distance threshold *maxdist* have been found for the Distance Filter. The local solver will be invoked only for the trial solutions that pass the two filters.

Analyzing the results from the local solver for all local solutions obtained is done in order to determine whether the solution is feasible, whether the solver stopped successfully, and to learn details about the solution or solution process that can turn out to be useful for the global search. Based on these observations the list of local solutions is updated. The Lagrange-multiplier vector is also updated to satisfy the requirements that all penalty weights should be larger than the largest Lagrange-multiplier found.

After every local search phase, the filter parameters are also revised. The Merit Filter *threshold* value is tightened if the starting trial solution point has a 'better' P_1 value (for the definition of P_1 see (3.6)) than *threshold* (which must be the case to pass the Merit Filter). The distance threshold *maxdist* is recorded for a new local solution. If the local solution has already been found, its distance threshold has to be updated for the new longest distance between a starting point and the local optimum.

When a trial point fails the Merit Filter test for *waitcycle* consecutive iterations, the threshold is relaxed by the *thfact* factor and the counter for failures is reset. Motivation for this originates from the idea that through the knowledge OptQuest develops about the problem, it will recommend better and better trial solutions. Some of the recommended points can have such high quality that the threshold determined by them will prevent trial points in other regions of the problem domain, which may belong to different region of attraction, from serving as starting points for the local search phase. The periodic relaxation is also consistent with the Tabu list idea in the meta-heuristic search framework.

5.4 Algorithmic Considerations

The above pseudo-code describes the main ideas of the OQGRG algorithm. However, any implementation has to deal with important details that will influence the usefulness, efficiency, performance, and applicability of the algorithm. This section deals with details like the treatment of linear and nonlinear constraints, the handling of integer variables, implementations of the filters, specifics of the GRG calls and the analysis of their results, accuracy and other considerations.

5.4.1 Linear and Nonlinear Constraints

OptQuest handles linear and nonlinear constraints very differently. Linear constraints are (optionally) satisfied by all trial points suggested by OptQuest. This is accomplished by solving an LP that finds the point that satisfies the linear constraints and is closest (using the L1 norm) to the newly generated, infeasible point. (For OptQuest to do this, entering linear constraints is part of setting up the problem.)

The situation is different with nonlinear constraints (*requirements* in OptQuest terms). They are evaluated by calling a user provided function, which also returns the objective value. When an evaluation is returned, OptQuest compares the *requirement* values to their bounds and penalizes the trial point by the maximum absolute violation weighed with a factor. The ranking of points is

based upon these penalized values. (See Chapter 3 for more discussion on OptQuest.)

Linear constraints are separated and passed to OptQuest in OQGRG to take full use of OptQuest's capabilities. Setting up the algorithm this way ensures that all trial points that are generated by the global search phase already satisfy the linear constraints.

In contrast with OptQuest, the local GRG solver handles both kinds of constraints similarly, so the problem passed to it is the original problem with all constraints. Lsgrg2, as a gradient-based GRG solver, first aims to achieve feasibility before attempting to find the optimal solution. It complements OptQuests weakness in finding trial solutions that satisfy the nonlinear constraints.

5.4.2 Filters

The two filters accomplish the goal of selecting a subset of trial solutions that have the best potential to quickly lead to different, better local optima. The Merit Filter selects only 'high quality' trial points that already have low merit function values, while the Distance Filter increases the probability that the selected trial points are distant enough from existing local solutions, i.e. promotes diversity among the selected trial points.

The *threshold* value for the Merit Filter is set after the first stage, when the 'pure OptQuest' iteration part of the algorithm ends. At that time a trial solution is selected as the best candidate for the local search. There are 2 ways to select the 'best' trial point: OptQuest uses the ranking method based upon a factor times the maximum violation added to the objective value. Alternatively, OQGRG can use a merit value function, the penalized objective that uses the exact penalty function added to the objective. The weights are chosen to be a factor times the largest available Lagrange multipliers if they are available, or sufficiently large default values as surrogates. Experiments were inconclusive in determining the use of which penalty gives better results.

When a trial point is accepted by the merit filter, *threshold* is decreased by setting it to the P_1 value of that point. If no better trial solution has been found for *waitcycle* iterations, the value is relaxed according to the following formula:

$$threshold_{new} := threshold_{old} + thfact * (1.0 + abs(threshold_{old})),$$

where the default value of *thfact* is 0.2 and that for *waitcycle* is 20. The additive 1.0 term is included so that *threshold* increases by at least *thfact* when its current value is near zero.

The Distance Filter helps insure that the starting points for the local search phase are diverse, in the sense that they are not too close to any previously found local solution. Its goal is to prevent the local solver from starting more than once within the basin of attraction of any local optimum. When a local solution is found, it is stored in a linked list, ordered by its objective value, as is the Euclidean distance between it and the starting point that led to it. If a local solution is located more than once, the maximum of these distances, *maxdist,* is updated and stored. For each trial point, *t,* if the distance between *t* and any local solution already found is less than *distfactor*maxdist,* the local solver is not started from the point, and we obtain the next trial solution from OptQuest.

This distance filter implicitly assumes that the attraction basins are spherical, with radii at least *maxdist.* The default value of *distfactor* is 0.75, and it can be set to any positive value. As *distfactor* approaches zero, the filtering effect vanishes, as would be appropriate if there were many closely spaced local solutions. As it becomes larger than 1, the filtering effect increases until eventually the local solver will never be started.

The combined effect of these 2 filters is that GRG is started at only a few percent of the OptQuest trial points, yet global optimal solutions are found for a very high percentage of the test problems. Some insight is gained by examining Figure 5.1, which shows the stationary point at the origin and the 6 local minima as dark squares for the 2 variable six-hump camelback function defined in (3.1),

labeled with their objective value. The ten points from which OQGRG starts GRG are shown as white diamonds.

The local minima occur in pairs with equal objective value, located symmetrically about the origin. There were 144 trial points generated during the iterations of Stage 1, and these, plus the 10 points in the initial population, are shown in Figure 3.4 . The best of these 154 points is the population point (0,0), so this becomes the first starting point for GRG. This happens to be a stationary point of F, so it satisfies the GRG optimality test (that the norm of the gradient of the objective be less than the optimality tolerance), and GRG terminates there. The next GRG start is at iteration 201, and this locates the global optimum at (.0898, -.7127), which is located twice. The other global optimum at (-.0898, .7127) is found first at iteration 268, and is located 6 times.

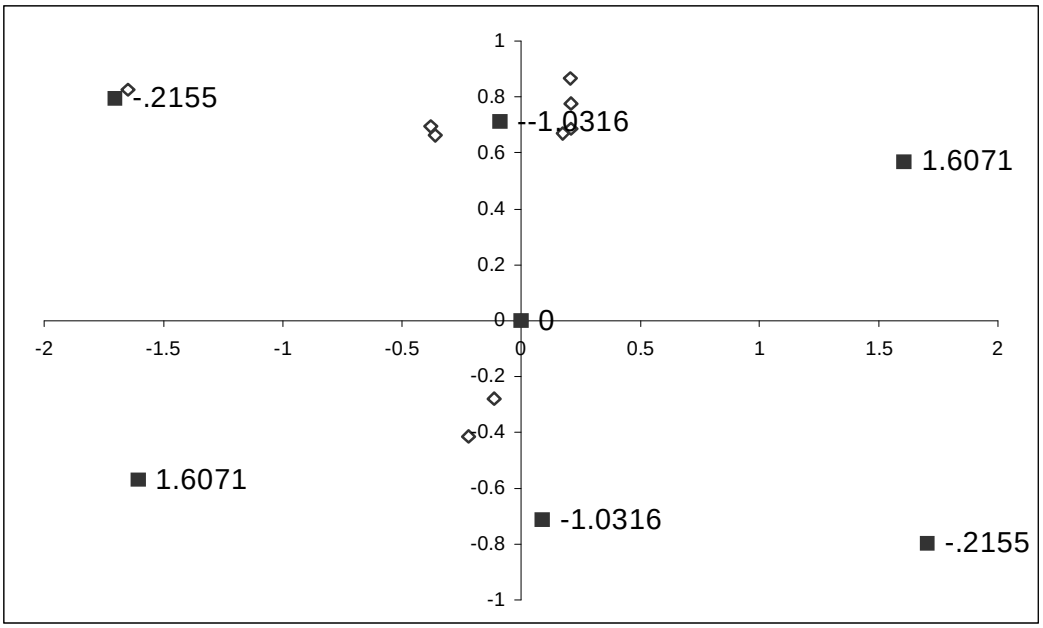


Figure 5.1: The 6 Local Optima and 10 GRG Starting Points for 6 Hump Camelback Function

The limit on total OQGRG iterations in this run is 1000. GRG is started at only 9 of the 846 OptQuest trial points generated in the main iterative loop. All but 2 of the starting points are in the basin of attraction of one of the two global optima. This is mainly due to the Merit Filter. In particular, the *threshold* values are always less than 1.6071, so no starts are ever made in the basin of attraction of the two local optima with this objective value. The merit filter alone rejects 498 points, the distance filter alone 57, and both reject 281.

Figure 5.2 illustrates the dynamics of the merit filtering process for iterations 155 to 407 of this problem, displaying the objective values for the trial points as white diamonds, and the threshold values as dark lines. All objective values greater than 2.0 are set to 2.0 for the sake of easier representation.

The initial threshold value is zero, and it is raised twice to a level of 0.44 at iteration 201, where the trial point objective value of -0.29 falls below it. GRG is then started and locates the global optimum at (.0898, -.7127), and the threshold is reset to –0.29. This cycle then repeats. Nine of the ten GRG starts are made in the 252 iterations shown in the graph. In this span, there are 12 points where the merit filter allows a start and the threshold is decreased, but GRG is not started at three of these because the distance filter rejects them.

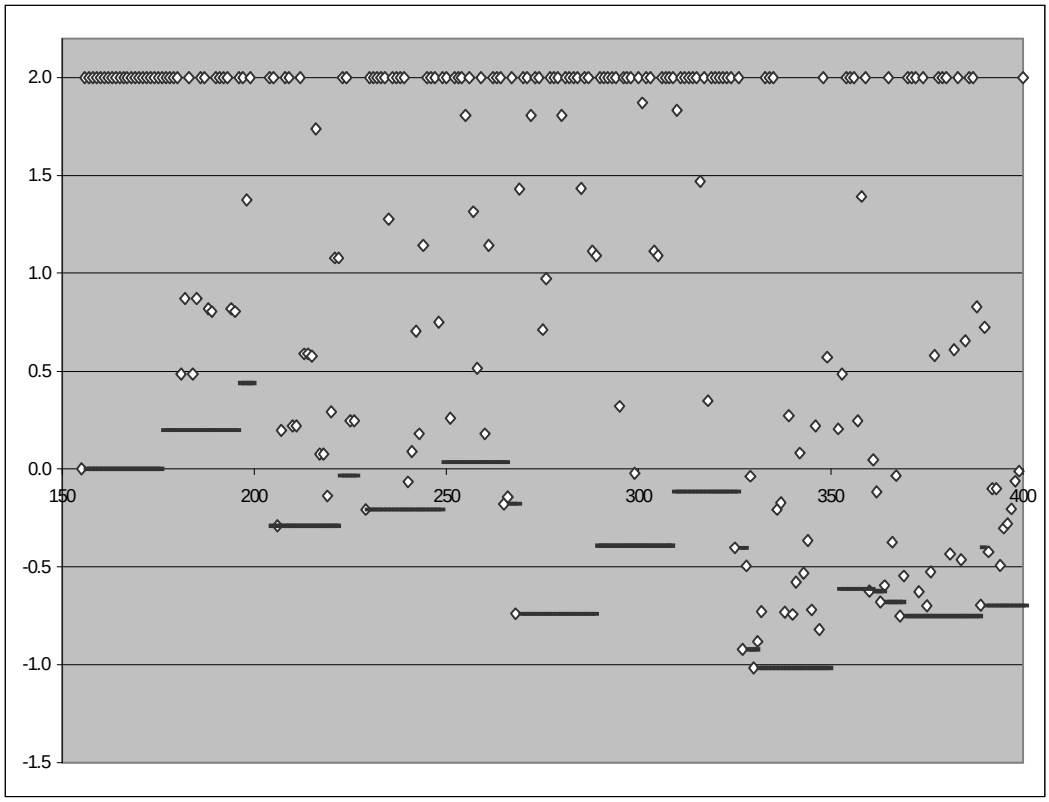


Figure 5.2: Objective and Threshold Values for Six-Hump Camelback Function for Iterations 155 to 400

Figure 5.3 shows the same information for iterations 408 to 1000. There is only one GRG start in this span. This is not due to a lack of high quality trial points: there are more good points than previously, many with values near or equal to –1.0310 (the global minimum is –1.0316), and the *threshold* is usually –1.0310 as well. Every time this *threshold* is raised, the Merit Filter accepts one of the next trial points, but 51 of the 52 accepted points are too near to one of the 2 global optima; thus, they are rejected by the Distance Filter.

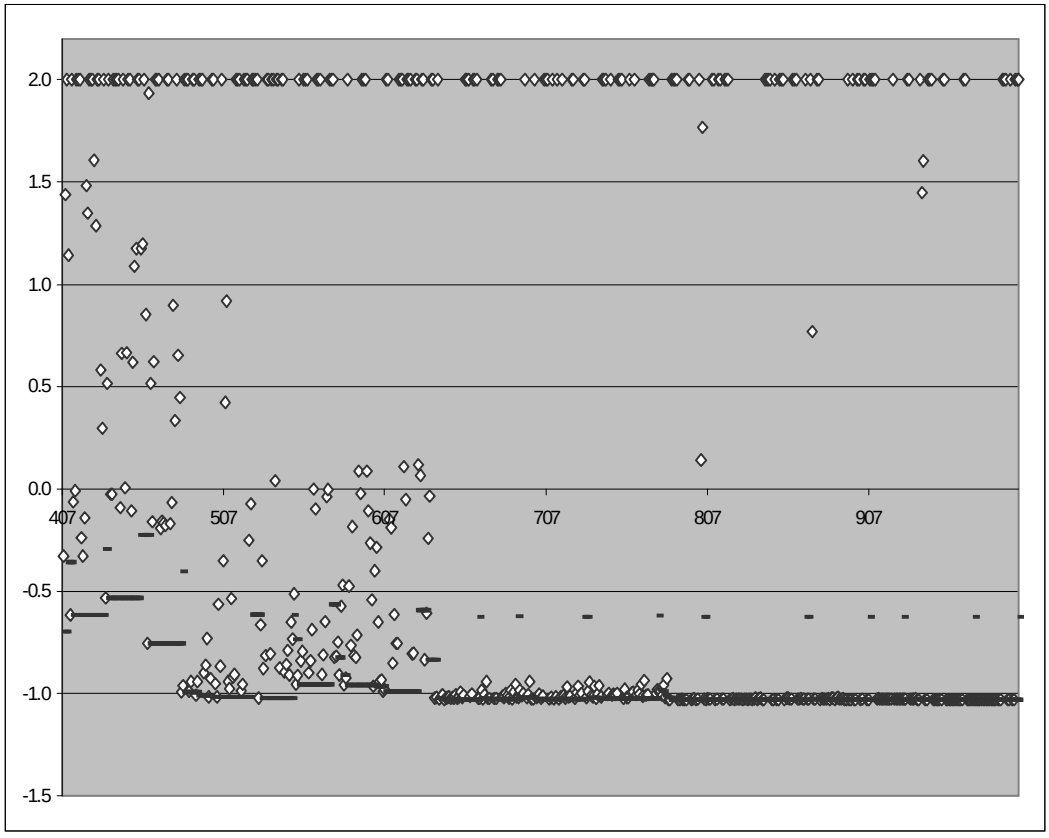


Figure 5.3: Objective and Threshold Values for Six-Hump Camelback Function: iterations 407 to 1000

This simple example illustrates a number of important points:

1. Setting the bounds on the variables to be too large in magnitude is likely to slow the OQGRG algorithm (or any search algorithm) and may lead to a poorer final solution. In the above example, if the variable bounds were [-2,2] rather than [10,10], the trial points generated by the initial population would

have had much better (lower) objective values. OptQuest can overcome this when the initial population is updated with 'better' points and the subsequent trial points are generated from this improved population.

2. GRG found a highly accurate approximation to the global solution of this unconstrained problem at its second call. OptQuest alone would have taken many more iterations to achieve this accuracy – or may not even have gotten to the neighborhood of the global solution. It can – and indeed, it did - happen in some runs, that after the discovery of an attraction basin, OptQuest myopically concentrates on area around that attraction basin. That area may not be in the neighborhood of the global solution's attraction basin. The diversification feature built into the algorithm will eventually try to explore other regions of the domain but it will likely take a large number of iterations to move to a new attraction basin.
3. The best trial point generated by the initial population may not have as good an objective value as those generated from the second or succeeding ones, especially if the variable bounds are too large. Using the best "first generation" point as the initial GRG starting point may not lead to as good a local solution as if some "second generation" points had been considered. For this reason our base case computational results use a first stage of 200 OptQuest trial points, which in this example would include all 144 first generation points and 56 from the second generation.

5.4.3 Integer Variables

OQGRG is designed to take advantage of OptQuest's capability of handling discrete variables. The remainder of this discussion will refer only to mixed integer problems since the vast majority of problems involving discrete variables actually uses integer or binary variables and transforming a problem with other kinds of discrete variables into integer or binary variables can take place at the modeling level.

Gradient based local solvers can deal with problems where the objective and the constraint functions are smooth, once differentiable functions with continuous first partial derivatives at all points in *S*. (See Chapter 5 for more details.) They cannot deal with discrete variables directly. Hence OQGRG fixes those variables before invoking the local optimizer.

Trial solutions with values for both continuous and integer variables are generated in the global search phase. OQGRG fixes the integer variables to their recommended values by putting the recommended value as both lower and upper bounds on these variables. Only the continuous variables can change during the local search. GRG solves the continuous part of the problem for fixed integer variables. As a result, any time the combination of the integer variables is different, GRG solves a different problem in the continuous variables.

If GRG determines that the problem is infeasible, it is only conclusive to the particular integer combination. This information is noted, although OQGRG

mainly relies on OptQuest to avoid recommending infeasible integer combinations; because of the penalty caused by the violation of some constraints, OptQuest will consider these trail points unattractive.

It is entirely possible that there is more than one local solutions for any different combinations of fixed integers. OQGRG maintains information for solutions with different integer values to accommodate the fact that the continuous sub-problems corresponding to these integer combinations are different. In the case of the Merit Filter, separate *threshold* values are maintained for different integer combinations. The updating schedule of these *threshold* values based on *waitcycle* unsuccessful trial points takes into account only trials with the same integer combinations.

Only local solutions with the same integer combination as the trial solution are considered when applying the Distance Filter. No additional information beyond the list of integer variables is necessary since all local optima have their *maxdist* parameter already associated with them.

5.4.4 GRG Calls and Analysis of GRG Results

Local searches (intensification phases, using the meta-heuristic terminology) are performed by invoking GRG. Although relatively much more 'expensive' (computationally intensive) than the global search, GRG is efficient in

finding locally optimal solutions with high degrees (4 to 8 decimal digits) of accuracy. Trial solutions that pass both the Merit and Distance Filters serve as starting points. The starting points do not have to be feasible, although OptQuest recommends trial points that are feasible at least in the linear constraints. Phase 1 of GRG attempts to find a feasible solution to the problem and Phase 2 starts from that point to find the optimal solution.

GRG can terminate under several different conditions. The three main reasons for termination are finding an optimal solution, finding the problem infeasible, and encountering some error during the effort to solve the problem.

A newly found optimal solution (the Kuhn-Tucker Conditions are satisfied, or no significant improvement has been made for a number of subsequent iterations) is placed into an ordered chained list, its location determined by the objective value of the solution. Parameters, such as the iteration number and the GRG call when the solution was found, the L_2 distance *maxdist* between the starting point and solution, and the merit function value at the starting point are recorded in addition to the variable and constraint values. If a solution is found more than once, parameters like the solution counter, best starting point's merit function value, and *maxdist* are updated.

GRG can conclude that it could not find a feasible solution. The problem could be truly infeasible, especially when there are integer variables, and those variables have been fixed for GRG. It is also possible, that for an otherwise

feasible problem Phase 1 of GRG failed to find a feasible point. In any case, OQGRG stores the stopping points for the solution effort, but these stopping points have to be handled carefully. Parameters, such as the iteration number when it was found, the GRG call when it was found, the L_2 distance *maxdist* between the starting and stopping point, and the merit function value at the starting point are recorded along with the stopping point.

Just as the already found local solutions, these infeasible points are used when the Distance Filter is applied to a trial solution. The trial point fails the test only if its distance from the infeasible solution point is less than *infeas_distfactor* * *maxdist*, where *infeas_distfactor* is 0.2, much smaller, than the 0.75 value of *distfactor* for feasible local solutions. By allowing to pass the Distance Filter a lot closer to a detected infeasible point, OQGRG allows for the possibility that a problem where GRG stopped at an infeasible point is actually feasible.

5.4.5 OptQuest and GRG Tolerances

OptQuest uses the absolute differences of values to control objective and variable precision (see Chapter 4), so the scaling of the problem can have a significant effect on the quality of the global search process. OQGRG uses 10^{-4} as default values for both objective and variable tolerances. Given two vectors of variables x1 and x2, if the infinity norm of (x1-x2) is less than 10^{-4}, then x1 and x2 are considered to be equal, and only one will appear in the OptQuest database of

trial points. Similarly, an objective improvement less than the objective precision is considered to be zero.

GRG controls the quality of the solutions through its feasibility and optimality tolerances (see (5.4) and (5.5)). These controls are relative measures, so the quality of solutions is less sensitive to the scaling of the problem. The value 10^{-4} is used as default for both tolerances in OQGRG.

To stay consistent with OptQuest's absolute measure of objective precision the Merit Filter also considers two penalized objective values different if the absolute difference between them is at least the objective precision. OQGRG uses OptQuest's variable precision to distinguish between two points. Local solutions are considered separate if the difference between at least one of their decision variables is larger than the variable precision. The selection of absolute measures to distinguish between two solutions can occasionally result in recording the same local solution more than once since the relative tolerances used in GRG's stopping criteria can cause the absolute differences to be larger than the precision default values.

5.4.6 Other GRG Options

The final implementation of OQGRG contains several options that were investigated during the algorithm development. Turning these options on or off can be controlled through the settings file, which is shown in Appendix A.

Returning local solutions to the population

OQGRG, in its default configuration, does not communicate local solutions found by GRG to OptQuest. Although it may seem that letting OptQuest know about the local solutions would have a positive effect on the global search, this need not to be true. The reason is that local solutions found by GRG typically have very good values compared to the trial solutions generated by OptQuest. Since these are almost always infeasible for the nonlinear constraints, the penalized values for the trial solutions will almost certainly be significantly degraded by the violations of these nonlinear constraints.

Very good points returned to OptQuest result in an intensive search around that good point at the expense of diversification. The global search process can fall into the trap of spending most of its effort in searching for good points around the singular very good point and will spend less effort in spanning out to search distant regions of the problem domain.

Using only nonlinear constraints

OQGRG takes advantage of OptQuest's ability to propose trial points that satisfy the linear constraints. This advantage comes at a price, though: a linear program has to be solved for every point that does not satisfy the linear constraints to map it into a feasible trial solution. This can significantly increase the effort spent in the global search phase to arrive at trial solutions that are feasible in the

linear constraints (although they can be still infeasible in the nonlinear constraints).

From an implementation point of view, the GAMS environment makes it possible to separate linear and nonlinear constraints. In other environments this possibility may not be readily available. OQGRG can still be used if the list of linear constraints is not available by assuming that all constraints are nonlinear.

Using normalized distance measure

The distance measure used in the Distance Filter is the Euclidean distance measure on the un-scaled variables. Alternatively, a normalized distance measure can also be used. The normalized distance between two points can be calculated as

$$dist^{norm}(x, y) = \sqrt{\sum_{i=1}^{n} \left(\frac{x_i - y_i}{ub_i - lb_i} \right)^2} \qquad (5.5)$$

where ub_i and lb_i are the upper and lower bounds on decision variable i. When using the normalized distance measure, *maxdist* and the trial points' distances from the local solutions are all calculated using (5.5). Accordingly, distance filtering takes places in the scaled variable space.

Although it seems reasonable to use normalized distances all the time, it has to be noted that OptQuest applies the precision calculations on un-scaled variables and thus not using normalized distances is more consistent with

OptQuest's approach to precision. Tests run on the Floudas example set did not show any significant difference in performance when the normalized distance measure was used.

Performing an initial GRG call – continuous variables

One way to improve the global search phase in Stage 1 is to place a very high quality point into the initial population. It can improve the search without affecting the diversity of points generated by the scatter search.

OQGRG can optionally perform an initial GRG call before any of the search takes place. Thus the OQGRG algorithm has an (optional) additional Step 2a between Steps 2 and 3:

```
2a. Call_GRG (starting point, local solution);
```

The *starting point* to the initial GRG call can be either user-provided or random. The resulting locally optimal solution can be used as a seed to build scatter search's initial reference set. The initial population will be built according to the same rules as described in Chapter 4, and it will have the same diversity in the population in addition to a very high quality point.

The effects of placing a locally optimal point into the initial reference set are discussed in Chapter 6.6.

Performing an initial GRG call – discrete variables

The initial GRG call for problems with discrete variables can handle the discrete variables in 2 different ways. OQGRG can either fix the discrete variables at their initial integer values and let GRG solve the nonlinear problem for these fixed values, or the relaxed problem can be passed to GRG. In the latter case the local solver solves the problem with the integer conditions relaxed, allowing the discrete variables to be continuous between their bounds. If the solution has all the discrete variables taking integer values, a local solution for the original problem has been found.

If the discrete variables do not take integer solutions, they are rounded to their nearest integer value and that point is passed as a recommended point to build the initial reference set for the Scatter Search. This point with rounded values may not be feasible but it is of no major concern: OptQuest will try to make it feasible in the linear constraints. As for the nonlinear constraints, OptQuest uses penalties to rank-order the different points, and GRG is very effective in achieving feasibility for the trial solution it is started from.

Determining the length of Stage 1

In Stage 1 only the global phase of the search is active, i.e. trial solutions from subsequent OptQuest iterations are not evaluated and returned to OptQuest without being subject to the Merit and Distance Filters. At this stage the Scatter

Search algorithm builds a ‘knowledge’ of the problem. OQGRG has two ways to control the length of this stage. The first method is let this phase run for a fixed number of iterations. In the majority of tests this number is selected to be 200. The rationale for the value 200 originates from the population size of 10. There are 45 distinct pairs of points, and for each pair 2 to 5 trial points are generated along their connecting line segments. (See Chapter 3 for details on how these points are generated.) The total number of trial points from the initial population thus ends up being a bit short of 200. (In fact, the total number of trial points depends on the relative quality of the points, the feasibility of the generated points and the effectiveness of mapping infeasible points into feasible ones.)

The second way of controlling the length of Stage 1 is to recognize when OptQuest regenerates the reference set after a cycle of its recommended trial solutions have been evaluated for the previous reference set. The new population will hold the best points found up to that point. The early trial solutions of the next cycle will originate from combinations of the ‘best’ points of the new population. It is efficient to start Stage 2 at this point. As a generalization of this idea, it is possible to count several cycles of trial points before starting Stage 2.

OptQuest solving only the discrete sub-problem

One OQGRG option implements a unique handling of the discrete variables and the linear constraints in which those variables appear. When this option is selected, OQGRG passes a projected problem of reduced size to OptQuest. This projected problem has only discrete variables and the constraints in which these discrete variables appear linearly. Its objective is the optimal value of the original problem objective over the continuous variables, with discrete variables fixed. This is called the *projection* of the original problem onto the space of the discrete variables. This projected problem can be a significantly smaller problem than the original one and OptQuest can very efficiently come up with trial solutions that contain the discrete variables satisfying the constraints.

The trial solutions are augmented to have all variables before they are passed to the local solver and the local solver solves the original problem with the discrete variables fixed as they are recommended in the trial solutions. Every trial solution is augmented and then solved by GRG in this option, thus there is no Stage 1 where the Scatter Search would learn about the problem domain without using the power of the local solver. The learning process takes place as the local search finds good local solutions.

As a result, the GRG iteration where the best value is found typically comes later than in the base case where a Stage 1 learning phase is performed before any local search takes place. On the other hand, OptQuest's effort is less

since it has to process a much smaller problem, and the information returned to it by the local solver (the optimal objective value over the continuous variables) is of much higher quality than in the base case (the penalized objective value at OptQuest's trial point). For computational results, see Chapter 6.6.3.

5.5 Implementation

OQGRG is implemented in ANSI compatible C. The code has been compiled under Microsoft Visual C++, version 6.0. OQGRG can be called as a function with the problem parameters passed in the argument list. The argument list is compatible with that of GRG, with one additional array to store the information about discrete variables. Algorithm parameters can be controlled through a settings file. Any parameter not referred to in the settings file takes its default value.

5.5.1 Interfaces

The OqGrgSub() calling function

The OQGRG algorithm has been implemented as a callable C-language function. The prototype for the OqGrgSub() callable interface is given in Table 5.2 below:

```
int OqGrgSub(LsgrgInfo *_lsinfo, long nvars_in, long nfun_in, long
             nobj_in, long maximize, long lvars[], double blvar[],
             double buvar[], double blcon[], double bucon[],
             double xx[], double fcns[], double rmults[], long
             nonbas[], double redgr[], long inbind[],long *nbind,
             long *nnonb, P_GCOMP p_user_gcomp, P_PARSH
             p_user_parsh, long nnz_user, long discrete[])
```

Table 5.2: Prototype for OqGrgSub()

Some of the parameters carry input data that describe the problem: a pointer to a (typedef LsGrgInfo) structure (*_lsinfo*) containing GRG and OQGRG algorithm parameters in addition to some parameters describing the problem; integer (*long*) types, such as number of variables *(nvars_in*), number of functions (*nfun_in*), index of the objective function (*nobj_in*), number of non-zero Jacobian elements (*nnz_user*), sense of optimization (maximize), and arrays marking the purely linear (*lvars[]*) and discrete (*discrete[]*) variables; floating point (*double*) types, such as arrays of variable starting points (*xx[]*), upper and lower bounds on variables (*buvar[]* and *blvar[]*) and constraints (*bucon[]* and *blcon[]*). There are pointers to the (typedef P_GCOMP) user function evaluation routine (*p_user_gcomp*) and the (typedef P_PARSH) user function which evaluates derivatives (*p_user_parsh*).

Output parameters are the integer (*long*) types such as number of binding constraints (*nbind*) and number of nonbasic variables (*nnonb*), and arrays of *doubles* for final function values (*fcns[]*), final solution values (*xx[]*), final multipliers (*rmults[]*), final reduced gradients (*redgr[]*), indices of non-basic variables (*nonbas[]*), and indices of binding constraints (*inbind[]*).

In this form, the user supplies a C function (typedef P_GCOMP) that evaluates the objective and constraint functions (*p_user_gcomp*), and an optional routine (typedef P_PARSH) that evaluates their first partial derivatives (*p_user_parsh*). In case the latter function is not provided, finite difference

approximations are used. A user written calling program supplies the problem size, bounds, and an initial point, and invokes the algorithm.

GAMS Interface

OQGRG has an interface between the C implementation and the GAMS algebraic modeling language (see www.gams.com), using C library routines generously provided by GAMS Development Company. The user function routine is replaced by one that calls the GAMS interpreter, and a special derivative routine accesses and evaluates expressions developed by GAMS for first derivatives of all nonlinear problem functions. GAMS identifies all linear terms in each function, and supplies their coefficients separately, thus identifying all linear constraints. This makes it possible to invoke the OptQuest option that maps each trial point into a point that satisfies the linear constraints. The derivative information supplied by GAMS also significantly enhances the performance of gradient-based NLP solvers, since only non-constant derivatives are re-evaluated, and these are always available to full machine precision.

Part of the motivation for developing the GAMS interface was the existence of a large set of global optimization test problems coded in GAMS, described in [Floudas et. al., 1999]. Test results for these problems are discussed in Chapter 6.

The oqalgpar.stg settings file

Algorithm parameters and options are communicated to the algorithm through an options text file. The options file with a brief explanation of the parameters is given Appendix A.

5.5.2 Data Structures

Local Solutions

Local solutions are stored in an ordered linked list of structures. The order of a structure in the list is determined by the objective value of the local solution it represents. Memory is dynamically allocated for the structures as new solutions are found. Objective value, decision variables, threshold value for the integer combination in the solution, and maximum distance traveled from a starting point to the local solution are stored in the structure along with some statistics, such as the iteration where the solution was first found, number of times the solution was found, and elapsed time until the solution was first found.

Algorithm Parameters

Algorithm parameters are placed into the info structure *OqGrgInfo* that is part of the larger *lsinfo* structure (see description of *lsinfo* and the settings file, oqalgpar.stg, in 5.5.1). A pointer to *lsinfo* is passed to OqGrgSub() and initialization, allocation of memory, and reading of the parameter values from the

settings file takes place early in the function. The memory for *OqGrgInfo* is freed before exiting OqGrgSub().

Trial Solutions

In the basic version of OQGRG trial solutions are returned from OptQuest in an array of floating point variables (*doubles*). The evaluated objective and nonlinear constraints are stored in another array of floating point variables. (The objective is always supposed to be the first element in the 0-based array.)

Determining when a trial solution generation cycle in Stage 1 ends requires a more complex procedure. Trial solutions are pulled, evaluated, and stored in a linked list. The penalized objectives and the number of the trial solutions are saved along with the decision variables. When all trial solutions generated from the current population have been obtained, OptQuest returns a flag. At this time OQGRG returns the evaluated points to OptQuest.

5.6 Outputs

Several output files can be generated during the solution process. They can provide different detailed information about the local solutions, the solution process, the reference set, the trial solutions, GRG calls, and filter decisions.

5.6.1 Iteration Logs

Output files, with .itn extension after the problem name, list the problem parameters, followed by one line per GRG call with the objective value, terminating condition, number of infeasible constraints, major GRG iterations, function evaluation calls, initial and final values of decision variables. The last section of the file lists the local solutions found ordered by the objective value. An example file for the six-hump camelback function, EX8_1_5.itn, is in Appendix B.

A list of objective values, Merit Filter results, penalized objective values, prevailing threshold values, Distance Filter results for every trial point is placed into files with the problem name followed by the .oqg extension. For iterations where GRG is called, objective of the GRG result, the termination condition and sum of infeasibilities are given. A summary of the filter activity and the list of local solutions found are placed at the end of the file. A shortened version of example file for the six-hump camelback function, EX8_1_5.oqg, is in Appendix C.

GAMS creates its own listing file, called the problem name followed by the .lst extension. The best local solution found by OQGRG is displayed in this output. For the solution summary part of the listings file EX8_1_5.LST for the six-hump camelback problem generated by GAMS see Appendix D.

Trial points with their evaluated objective and constraint values can optionally be outputted into the trialpts.log file. Similarly, the active population with the constraint values belonging to the member points can be followed through the poplist.log file. Activation of these file outputs can be controlled through the oqalgpar.stg settings file.

5.6.2 Other Output Files

OQGRG can provide a file that lists detailed information for every GRG call. This file has the file name followed by the .log extension. The extent of details can be controlled with the GRG print option parameters through the oqalgpar.stg settings file.

Details from the OQGRG algorithm can be printed into a file with the problem name followed by the .tmp extension. The level of details can be controlled by the OQGRG_DEBUG value in the oqalgpar.stg file.

Information from the GAMS interface can be followed in the file with the filename followed by the .tst extension.

6. Computational Results

This chapter focuses on testing OQGRG on a large set of problems. The problems presented vary greatly in the number of variables and constraints; some are linear but most are non-linear. There are a few with discrete variables. Many have multiple local optima. In most cases OQGRG performs well, actually quite well, in its effort to find the global (or at least the best known) optimum. In several cases it finds better solutions than the best known values. As it can be expected, there are a few problems where OQGRG runs into difficulties. However, changing the algorithm parameters from their default values usually results in OQGRG being able to solve those problems, too.

6.1 The Floudas Problem Set

Part of the motivation for developing the GAMS interface for OQGRG was the existence of a large set of global optimization test problems coded in GAMS, described in [Floudas et. al., 1999]. This text describes some problems that cannot be represented in GAMS, but there are many that can, and these can be downloaded from http://titan.princeton.edu/TestProblems/. Characteristics of these problems are contained in Table 6.1. There are 142 problems plus two groups of problems in all.

Series	problems	max vars	max discrete vars	max linear constraints	max nonlinear constraints	Problem Type
EX2_1_x	14	24	0	10	0	concave QP (min)
EX3_1_x	4	8	0	4	6	quadratic obj and constraints
EX4_1_x	9	2	0	0	2	obj or constraints polynomial
EX5_2_x	2	32	0	8	11	bilinear-pooling
EX5_3_x	2	62	0	19	34	distillation column sequencing
EX5_4_x	3	27	0	13	6	heat exchanger network
EX6_1_x	4	12	0	3	6	gibbs free energy min
EX6_2_x	10	9	0	3	0	gibbs free energy min
EX7_2_x	4	8	0	3	12	generalized geometric prog
EX7_3_x	6	17	0	10	11	robust stability analysis
EX8_1_x	8	6	0	0	5	small unconstrained, constrained
EX8_2_x	5	55	0	6	75	batch plant design-uncertainty
EX8_3_x	14	141	0	43	65	reactor network synthesis
EX8_4_x	8	62	0	0	40	constrained least squares
EX8_5_x	6	6	0	2	2	min tangent plane distance
EX8_6_1	N	3N	0	0	0	Lenard-Jones energy min
EX8_6_2	N	3N	0	0	0	Morse energy min
EX9_1_x	10	29	6	27	5	bilevel LP
EX9_2_x	9	16	3	11	6	bilevel QP
EX12_2_x	6	11	8	9	4	MINLP
EX14_1_x	9	10	0	4	17	infinity norm solution of equations
EX14_2_x	9	7	0	1	10	infinity norm solution of equations
Total:	142 + 2N					

Table 6.1: Floudas Problem Set Summary

Many problems arise from chemical engineering, but some are from general problem classes. Most are small, but over a dozen have more than a 100 variables and comparable numbers of constraints. 11 have both continuous and discrete variables. Almost all of the problems without discrete variables have local solutions distinct from the global solution, and the majority of problems have constraints. Sometimes all constraints are linear, as with the concave quadratic programs of series EX2_1_x. Many problems have nonlinear constraints, and these are often the sources of the non-convexities. For example, there are many problems arising from pooling and blending applications with bilinear constraints.

EX2_1_7 has results published for 5 parameter sets, those are separated into the 5 problems EX2_1_7_1 – EX2_1_7_5.

GRG has not been able to find a feasible solution for EX5_3_3 at all. The problem is suspected to have errors in its GAMS model.

For the robust stability analysis problems of EX7_3_x, the goal is to find the global minimum of the system's characteristic equation that has to be larger than 1 to prove that the system is stable. The models of EX7_3_4, EX7_3_5, and EX7_3_6 could be simplified from the published models. OQGRG could not find feasible solutions for EX7_3_5 and EX7_3_6 because GRG could not find any feasible solutions. This does not necessarily imply that the problem is infeasible; GRG may have located only a local minimum for the phase one objective. (In principle, OQGRG could be applied to the phase one problem as well; currently it

is not designed to do this.) The modified EX7_3_5_mod had a feasible solution, the best known value, greater than 1. Although a feasible solution for EX7_3_6_mod was found with both the 1gen and 2gen strategies (see later for details), the objective was 0.00, not achieving the best known value.

Some problems from the EX8_3_x reactor network synthesis models have been found incorrect as downloaded, and were modified after consultation with the authors of [Floudas et. al., 1999]. But even now EX8_3_12, EX8_3_14, and EX8_3_14_carl often return infeasible solutions only. The largest problems in this series have many local solutions. Finding the best know ones proved to be difficult. However, OQGRG performed quite well in finding solutions within a few percentage points of the best known solutions.

The symbol N in the rows for the series EX8_6_1 and EX8_6_2 is the number of particles in a cluster whose equilibrium configuration is sought via potential energy minimization. Each particle has 3 coordinates. A total of 6 coordinates for the first 3 particles are fixed, so there are 3N-6 variables for these problems.

EX8_6_1 is the Lennard-Jones energy minimization problem for heavy atom rare gas clusters. The test runs were performed on a slightly modified, unconstrained reformulation of the published model. EX8_6_2 is the similar Morse energy minimization problem for metal clusters. For both problems several distinct number clusters were included in the test runs. These are the

EX8_6_1R05 – EX8_6_1R30 and EX8_6_2R05 – EX8_6_2R50 series where the last 2 digits mark the number of particles in the cluster.

Some of the problems were formulated to accommodate special solver-imposed constraints, i.e. DICOPT does not allow integer variables to appear nonlinearly. EX12_2_3_N and EX12_2_4N have been modified (simplified) since OQGRG allows integer variables to appear nonlinearly.

The best known objective value and (in most cases) the corresponding variable values are provided in [Floudas, et. al., 1999]. These values are considered the optimum values when solution gaps are calculated for the test runs. Appendix E contains information about the size, types of variables and constraints, and the best known solution for the extended 163 problems. This extended list has both the original and the reformulated versions of 9 problems where it is possible to improve upon the original models.

6.2 The Base Case

This section describes the results obtained when the OQGRG algorithm is applied to continuous problems of the Floudas test set. The options and main algorithm parameters used are shown in Table 6.2.

OptQuest and OQGRG Parameters	GRG Parameters
Use linear constraints = yes	Feasibility tolerance = 1.e-4
Use initial GRG call = no	Optimality tolerance = 1.e-4
Total iterations = 1000	Consecutive iterations for fractional change termination = 6
Total stage 1 iterations = 200	
Waitcycle = 20	
Thfact = 0.2	
Distfact = 0.75	
OptQuest search type = *boundary*	
Boundary search parameter = 0.5	
OptQuest Variable Precision = 1.e-4	
OptQuest Objective Precision = 1.e-4	
Check for duplicates in database = yes	

Table 6.2: OptQuest, GRG, and OQGRG Parameters and Options Used

In the base case no initial GRG call is performed at the start of Stage 1. OptQuest's precision parameters are set to 10^{-4}, the database feature is turned on (so duplicate trial solutions are not generated) and all trial points are required to satisfy the linear constraints. The boundary search strategy is used with the default 0.5 parameter value, which directs the trial points generated towards the boundary

of the region defined by the variable bounds and general linear constraints 50% of the time. OQGRG runs a total of 1000 iterations of which the first 200 belong to Stage 1, where the trial points are not considered as starting points for GRG calls. The optimality and feasibility tolerances of GRG are also set to 10^{-4}, and GRG terminates if there are 6 consecutive iterations with a fractional objective change less than the optimality tolerance. Results obtained from running all of the problems with the base case can be found in the table of Appendix G. These computations were performed on an IBM compatible PC with a 450 MHz Pentium III processor and 320 Mbytes of RAM, running under Windows 98 OS.

Besides the best known value, the best objective value of the base case, the iteration when the best value was found, the GRG call when the best value was found, total GRG calls, function number when the best value was found, total function calls, time when the best value was found, total runtime, number of local solutions found, and the solution gap are reported for all problems.

The gap is reported as percentage and calculated by:

$$gap = \frac{f_{obj} - f_{best-known}}{abs\left(f_{obj} - f_{best-known}\right)} \quad \text{when } f_{best-known} \neq 0 \text{ or}$$

$$gap = f_{obj} - f_{best-known} \quad \text{when } f_{best-known} = 0 \text{ .}$$

132 of the 163 problems (81%) are solved with a gap of less than 1% or the absolute difference being less than 0.001 between the best known value and the

best objective found. There are 7 more problems where, although the gap is greater than 1%, it is assumed that the solution is actually the best known value. Of the 9 reformulated problems only 2 did not improve the results. In 102 problems (63%) the best value is found by the first GRG call, in an additional 11 (7%) the best value is found by the second GRG call, showing the effectiveness of the Scatter Search to find a good basin of attraction in the first 200 iterations or at least early in the search process.

More detailed analysis of the test runs, remedies for failed problems, and solution efforts where accuracy needed improvements are discussed below.

6.3 Results by Problem Size – Base Case, Continuous Problems

Table 6.3 shows statistics for 6 different size groups of the 150 continuous problems. Table 6.4 shows the same statistics for 145 problems, omitting the 4 notoriously non-solving problems for which the GRG calls end infeasible (EX5_3_3, EX7_3_5, EX7_3_6, EX8_3_14_carl). and one where there is some other problem (EX6_2_14, where the published best value is probably incorrect).

Variable Range	No. of Problems	No. of Probs Solved	Iterations to Best	GRG Calls to Best	Total GRG Calls	GRG Ratio	Locals Found	Function Call to Best	Total Function Calls	Function Ratio	Time to Best	Runtime	Time Ratio
1--4	36	34	222.3	1.5	7.8	0.20	2.2	253.3	1242.6	0.20	0.3	1.0	0.30
5--7	28	26	243.1	1.9	12.5	0.15	3.7	384.4	2434.4	0.16	0.6	2.4	0.26
8--12	26	21	277.5	3.7	17.6	0.21	7.5	1119.0	4123.9	0.27	1.4	4.1	0.34
13--22	22	16	238.5	2.4	26.6	0.09	6.9	461.4	4861.0	0.09	4.0	8.6	0.47
23--78	19	15	414.1	11.7	31.2	0.37	19.4	5083.6	15763.3	0.32	40.1	69.8	0.57
84--144	19	9	555.7	13.8	29.7	0.47	22.9	14059.3	30948.2	0.45	170.7	291.1	0.59
Overall	**150**	**121**	**305.1**	**5.0**	**18.8**	**0.26**	**8.9**	**2834.8**	**8118.9**	**0.35**	**27.9**	**48.7**	**0.57**

Table 6.3: Performance Statistics for 6 groups of problems (all)

Variable Range	No. of Problems	No. of Probs Solved	Iterations to Best	GRG Calls to Best	Total GRG Calls	GRG Ratio	Locals Found	Function Call to Best	Total Function Calls	Function Ratio	Time to Best	Runtime	Time Ratio
1--4	35	34	222.9	1.5	7.8	0.20	2.2	254.6	1246.6	0.20	0.3	1.0	0.30
5--7	28	26	243.1	1.9	12.5	0.15	3.7	384.4	2434.4	0.16	0.6	2.4	0.26
8--12	26	21	277.5	3.7	17.6	0.21	7.5	1119.0	4123.9	0.27	1.4	4.1	0.34
13--22	19	16	242.4	2.5	26.6	0.10	7.6	462.7	4652.5	0.10	4.3	8.8	0.49
23--78	18	15	383.9	9.7	30.2	0.32	20.5	5183.2	16443.2	0.32	41.3	72.7	0.57
84--144	18	9	575.4	14.6	31.3	0.47	24.2	14821.5	32604.2	0.45	178.7	303.3	0.59
Overall	**145**	**121**	**303.4**	**4.8**	**18.7**	**0.26**	**9.2**	**2900.3**	**8265.7**	**0.35**	**28.5**	**49.6**	**0.57**

Table 6.4: Performance Statistics for 6 groups of problems (reduced set)

The base case solved 121 (84%) of the 145 problems within 1% or with less than 0.001 absolute difference from the best known value. Solutions for an

additional 6 problems (EX6_1_3, EX8_4_6, EX8_5_3 – EX8_5_6) are so close to the published ones (although the gap is larger than 1%), that they can be considered solved to the best known values. In the base case, EX9_1_8 showed a saddle point as the best solution and the true global optimum was also found. Interestingly, every other algorithm option resulted in the correct optimum reported.

With these additional 7 problems, 128 (88%) of the 145 problems were solved to the best known value by the base case. The problems where the global solutions were not found belong to the linearly constrained quadratic problems (EX2_1_x), the Lennard-Jones and Morse energy minimization problems (EX8_6_x), and the chemical reactor network synthesis (EX8_3_x) problems. Ways to change algorithm parameters to solve these remaining problems to within 1% accuracy will be discussed in later sections.

The problems are grouped and ordered in terms of increasing number of variables, and the number of local optima found increases with problem size, with an average of 22.9 (24.2 for the reduced set) for the largest group. The measures of computational effort to find the best solution (iterations to best, GRG calls to best, function calls to best, and time to best) are all gratifyingly small, and most increase slowly with problem size. Function calls are much higher for the largest group (84 to 144 variables), reflecting the GRG effort required to solve these problems, which have many nonlinear constraints. Average total GRG calls are

fairly stable at between 17 to 31 over the last four groups, and do not increase rapidly with problem size. This further demonstrates the effectiveness of the distance and merit filters described in Chapter 5.

The ratio columns provide additional evidence that the best solution is found early in the iterative process. The smallest of these is the GRG ratio, which varies from 0.09 to 0.47, meaning that the best solution is found in the first 9% to 47% of GRG calls. This ratio is highly correlated with the function call ratio, because function calls due to GRG (all those over 1000) dominate as problem size increases. This implies that, for these problems, a criterion that stops OQGRG when the fractional change in the best feasible objective value found thus far is below a small tolerance for some (reasonably large) number of successive iterations, would rarely terminate the algorithm before the best solution was found.

6.4 Varying the length of Stage 1

Three values for the number of stage one iterations were tested for the 150 Floudas problems with no discrete variables. The base case had 200 initial iterations, as described above. A strategy, called the "*1gen*", has as many Stage 1 iterations as required to generate all "first generation" trial points (those created from the initial population) before the population is updated. Similarly, the "*2gen*" strategy has as many Stage 1 iterations as required to generate all first and second generation trial points: those created from the initial population and from the

subsequent, already once updated population. 1000 total iterations were used for all three strategies. Detailed, problem level information for the 1gen and 2gen cases can be found in Appendices G and H.

The averages for various measures of computational effort and achievement over all 150 problems for these three stage one strategies are shown in Table 6.5 below. The column headed "Probs < 1%", shows the number of problems solved to a gap of 1% or less by the strategy, while the next to last column gives the number of these successful runs where the best solution was found at the first or second GRG call.

Stage 1	Stage1 Iterations	Iterations to Best	GRG calls to Best	Total GRG calls	Function Calls to Best	Total Function Calls	Time to Best	Total Time	Locals Found	Solved in 1 or 2 GRG	Probs < 1%
1gen	162	249.9	4.3	20.3	2398.2	7871.4	21.7	44.3	9.0	72%	121
200	200	305.1	5.0	18.8	2835.0	8118.9	27.9	48.7	8.9	73%	121
2gen	316	399.8	4.5	17.8	2308.1	6372.4	24.6	40.5	8.2	70%	121

Table 6.5: Effects of Varying the Number of Stage 1 Iterations

The 3 different strategies for Stage 1 resulted in similar quality solutions. All 3 have solved 121 of the 150 problem within 1% tolerance. The best values were found in the first or second GRG call in 70% - 73% of the problems. The average number of local solutions found are close, between 8.2 and 9. The smaller

number of local solutions in case of the *2gen* strategy could be because of the Scatter Search more intensely focusing on good region of attractions.

1gen and *2gen* have a variable number of initial iterations due to OptQuest's decision to update the population when it concludes that no more good trial solutions can be generated. For the *1gen* case the average length of Stage 1 is 162 iterations, for *2gen* it is 316. It can be observed that as Stage 2 starts later, the iterations finding the best values also occur later. The average number of GRG calls to the best value found are roughly the same, between 4.3 and 5. Total GRG calls go down slightly as Stage 2 starts later. It could be because there are less overall Stage 2 iterations available to start GRG. Total function calls are close for the first 2 cases, but its value is quite lower for *2gen*. This might be because after updating the population twice already, the recommended trial solutions are of higher quality, requiring less GRG function evaluations.

Statistics for different size groups for the *1gen* and *2gen* strategies are in Tables 6.6 and 6.7.

Variable Range	No. of Problems	Iterations to Best	GRG Calls to Best	Total GRG Calls	GRG Ratio	Locals Found	Function Call to Best	Total Function Calls	Function Ratio	Time to Best	Runtime	Time Ratio
1--4	36	157.8	1.1	7.3	0.16	2.2	210.8	1248.0	0.17	0.3	1.0	0.26
5--7	28	242.7	3.1	16.6	0.19	4.1	460.8	2278.6	0.20	0.7	2.5	0.27
8--12	26	215.0	2.2	20.4	0.11	6.6	748.2	4145.0	0.18	1.0	4.4	0.23
13--22	22	201.7	1.8	28.0	0.06	10.1	312.0	5585.1	0.06	4.7	12.4	0.38
23--78	19	347.8	9.5	39.1	0.24	26.5	3859.5	21111.8	0.18	25.8	93.3	0.28
84--144	19	440.6	12.7	22.4	0.57	13.7	12610.1	23169.4	0.54	136.9	230.4	0.59
Overall	**150**	**249.9**	**4.3**	**20.3**	**0.21**	**9.0**	**2398.2**	**7871.4**	**0.30**	**21.7**	**44.3**	**0.49**

Table 6.6: Performance Statistics for 6 Groups of Problems, "1gen" Option

Variable Range	No. of Problems	Iterations to Best	GRG Calls to Best	Total GRG Calls	GRG Ratio	Locals Found	Function Call to Best	Total Function Calls	Function Ratio	Time to Best	Runtime	Time Ratio
1--4	36	298.5	1.6	9.1	0.18	1.9	370.0	1384.4	0.27	0.3	1.2	0.29
5--7	28	345.2	1.7	10.6	0.16	3.3	519.6	1887.2	0.28	0.8	2.2	0.34
8--12	26	412.0	5.3	21.0	0.25	7.6	1427.1	5223.0	0.27	1.7	4.9	0.35
13--22	22	361.3	2.0	25.5	0.08	8.5	484.0	4495.6	0.11	5.2	11.8	0.44
23--78	19	549.9	11.8	30.7	0.39	19.2	5843.9	13928.7	0.42	41.5	75.2	0.55
84--144	19	558.1	8.6	19.1	0.45	17.1	8736.6	19303.4	0.45	149.1	228.8	0.65
Overall	**150**	**399.8**	**4.5**	**17.8**	**0.25**	**8.2**	**2308.1**	**6372.4**	**0.36**	**24.6**	**40.5**	**0.61**

Table 6.7: Performance Statistics for 6 Groups of Problems, "2gen" Option

Some interesting observations can be made about the group with the largest problems. Total GRG calls and the GRG call to the best found value drop off a great deal for the *2gen* strategy. As it happens, the quality of solutions for these

problems is a lot worse in this case. By waiting for 2 updates in the population, Scatter Search focuses on some good regions, but these good regions do not contain the global optimum. The problems with the largest size have many local solutions. Scatter Search running too long on its own will myopically turn to a few regions it found promising, losing sufficient diversity in the search. For this largest size group with the 200 iteration, the Stage 1 case performs the best. More GRG calls result in more locals found with a better chance to find the global optimum.

The benefits of starting Stage 2 earlier are: (1) the best solution is often found earlier, since the first GRG call usually finds the best solution, and (2) trial points which would be skipped in a longer stage one are eligible to be GRG starting points, and can lead to good GRG solutions. Since the population loses diversity as it is updated by the aggressive update currently used in OptQuest, these missed opportunities may not recur before the population is reinitialized. The advantages of a longer Stage 1 are: (1) The best point found by OptQuest in a longer stage one should, on average, have higher quality than in a shorter one, which leads to somewhat better results on the first GRG call, and (2) these higher quality best points should have lower values for the exact penalty function, P_1, which becomes the initial value for the merit filter threshold. This lower value leads to fewer GRG calls in Stage 2. The number of GRG calls is also influenced by other factors, so the effect is not monotonic.

6.5 The Effects of Having an Initial GRG Call

If Stage 1 begins with an initial GRG call, 101 (68%) of the 150 continuous problems solved with a gap of less than 1% or the absolute difference being less than 0.001 between the best known value and the best objective found. (Of the 9 reformulated problems, only 1 did not improve the results.) This compares to 121 when no initial GRG call is made, so the performance of OQGRG is worse when an initial GRG call is made. This is surprising, and is discussed below. In 75 problems (50%) the best value was found by the first, initial GRG call, in an additional 38 (26%) by the second GRG call after the 200 iterations of the OptQuest search phase.

The reduction in the number of problems solved to a small gap can be explained by the significantly reduced number of GRG calls (see Table 6.8). The initial GRG call returns a local solution. This high quality point is used as a seed point to start the Scatter Search's reference set. The search process will be biased toward the region around this high quality point, resulting in a less diverse sequence of trial solutions. Additionally, when the Merit Filter's starting threshold value is selected at the end of Stage 1 of the OQGRG algorithm, it almost always is equal to the objective value found by GRG. Fewer trial points end up passing both the Merit and Distance Filters, fewer GRG calls are made and thus fewer regions of attractions are explored. Problems with large numbers of local solutions are affected the most: the average local solutions found in the group with 23-78

variables fell to 6.6, 1/3-rd the number of the base case. For the group with 84-144 variables, it fell to 10.9, less than half of the base case's 22.9.

The large reduction in the number of problems where the first GRG call finds the best solution (from 102 down to 75) and the significant increase of the second GRG call finding best solution (from 11 up to 38) is the evidence of the power of Scatter Search learning about the problem in the global search phase of Stage 1. Table 6.8 shows statistics for 6 different size groups of the 150 continuous problems, now with the initial GRG call option turned on.

Variable Range	No. of Problems	Iterations to Best	GRG Calls to Best	Total GRG Calls	GRG Ratio	Locals Found	Function Call to Best	Total Function Calls	Function Ratio	Time to Best	Runtime	Time Ratio
1--4	36	77.4	1.5	4.3	0.34	1.8	155.3	1200.3	0.13	0.2	1.0	0.20
5--7	28	136.6	2.3	7.6	0.30	3.3	913.6	2241.5	0.41	0.7	2.3	0.32
8--12	26	258.8	3.0	7.5	0.40	2.8	866.7	2613.6	0.33	1.0	3.3	0.32
13--22	22	128.3	1.8	7.9	0.23	2.9	301.0	1322.1	0.23	3.0	6.2	0.49
23--78	19	149.6	4.0	7.6	0.53	6.6	1522.9	4147.9	0.37	6.7	16.1	0.42
84--144	19	359.3	6.3	14.5	0.44	10.9	6468.3	14871.7	0.43	49.8	120.7	0.41
Overall	**150**	**172.6**	**2.9**	**7.7**	**0.37**	**4.2**	**1421.2**	**3776.5**	**0.38**	**8.0**	**19.6**	**0.41**

Table 6.8: Performance Statistics for 6 Groups of Problems with Initial GRG Call

Along with the total GRG and function calls, the iterations, GRG calls, and function calls to find the best values are all lower than for the base case. The GRG

and function calls do not increase with the problem size as rapidly as in the base case so the ratio values are more balanced over the problem size.

Relaxing some of the OQGRG parameters can balance the effect of the initial GRG call. Table 6.9 shows the results when *waitcycle* is reduced to 10 (from 20), and *thfact* is increased to 0.3 (from 0.2).

111 (74%) of the 150 problems is solved to within 1% or 0.001 of the best known values. GRG finds the best solution 75 times at its first call and 33 times at its second call. The average number of GRG calls to best solution found is up to 4.3 and the function calls till the best solution found have correspondingly grown to 1870. Total GRG and functions calls as well as the locals found are also higher.

Variable Range	No. of Problems	Iterations to Best	GRG Calls to Best	Total GRG Calls	GRG Ratio	Locals Found	Function Call to Best	Total Function Calls	Function Ratio	Time to Best	Runtime	Time Ratio
1--4	36	64.8	1.4	4.8	0.29	1.8	141.3	1226.0	0.12	0.2	1.0	0.20
5--7	28	134.6	3.1	13.0	0.24	4.8	1036.0	3356.8	0.31	0.8	2.9	0.29
8--12	26	228.7	3.0	11.8	0.25	4.2	703.1	4647.8	0.15	1.0	4.4	0.22
13--22	22	136.6	3.2	14.2	0.23	5.4	441.0	1546.4	0.29	3.8	9.5	0.40
23--78	19	221.1	8.5	14.4	0.59	13.3	3432.7	6795.5	0.51	14.3	28.4	0.50
84--144	19	466.9	10.7	28.4	0.38	22.6	8495.2	22493.4	0.38	77.4	159.4	0.49
Overall	**150**	**185.4**	**4.3**	**13.0**	**0.33**	**7.3**	**1870.1**	**5541.8**	**0.34**	**12.1**	**25.8**	**0.47**

Table 6.9: Performance Statistics for 6 Groups of Problems with Initial GRG Call, Relaxed OQGRG Parameters

These results are closer to the base case, but the total GRG calls, locals found and runtime are still less than half, while the total function calls are about two-thirds of the values observed for the base case. There is considerably less effort spent to solve the problems while the number of problems solved is down to only 111 from 121.

6.6 Problems with Discrete Variables

OQGRG can exploit OptQuest's efficient handling of discrete variables and thus solving MINLPs. The integer variables can appear nonlinearly. The following sections discuss the test run results on the problems with integer variables.

6.6.1 The Base Case

Table 6.10 contains results of solving the 13 problems in the Floudas test set which have discrete variables. The problems are sorted first by number of discrete variables, then by number of all variables.

Name	Best Known Solution	N0. of Variables	No. of Discrete Vars	Linear Constraint	Nonlinear Constraints	Objective	Best Iteration	Best GRG Call	Total GRG Calls	Total Enum	Locals Found	Function to Best	Total Function Calls	Time to Best	Runtime	GAP
EX12_2_2	1.08	3	1	2	1	1.08	201	1	8	2	1	208	1057	0.3	0.9	0.0%
EX12_2_1	7.67	5	3	3	2	7.67	249	4	15	8	7	271	1112	1.6	4.2	0.0%
EX12_2_6	-17.00	5	3	4	1	-17.00	201	1	15	8	5	202	1205	1.0	3.6	0.0%
EX9_2_9	2.00	12	3	11	1	2.00	201	1	8	8	3	202	1035	1.9	7.6	0.0%
EX12_2_3_N	4.58	7	4	5	4	4.58	848	22	26	16	11	1313	1632	4.2	4.9	0.0%
EX12_2_3	4.58	11	4	9	4	5.27	786	13	20	16	8	1027	1454	5.9	6.9	15.1%
EX9_1_9	3.11	17	5	16	1	3.11	201	1	15	32	10	202	1071	9.0	37.7	0.0%
EX12_2_5	31.00	8	6	9	1	31.00	201	1	10	64	4	202	1078	5.5	22.5	0.0%
EX9_1_6	-49.00	20	6	19	1	-49.00	201	1	37	64	33	202	1083	18.8	77.7	0.0%
EX9_1_7	-26.00	23	6	21	1	-26.00	227	4	46	64	34	261	1341	38.3	74.7	0.0%
EX9_1_3	29.20	29	6	27	1	-29.20	201	1	66	64	61	202	1230	34.2	142.7	0.0%
EX12_2_4	-0.94	11	8	4	3	-0.91	202	2	26	256	7	265	1845	12.0	16.5	3.1%
EX12_2_4N	-0.94	11	8	4	3	-0.94	240	9	24	256	8	266	1071	14.0	14.2	0.0%
Averages		**12**	**5**	**10**	**2**		**305**	**5**	**24**	**66**	**15**	**371**	**1247**	**11.3**	**31.86**	

Table 6.10: Solution Statistics for 13 Problems with Discrete Variables

All problems are solved to very small gaps except EX12_2_3 and EX12_2_4, with gaps of 15.1% and 3.1% respectively. Increasing the number of iterations to 5000 or 10,000 did not yield better solutions for these problems. Their reformulated versions, EX12_2_3_N and EX12_2_4N, solved with no trouble. (See the reformulated models later.) The column headed “total enum” contains the number of GRG calls needed to solve the problem by complete enumeration of all integer combinations. The total number of GRG calls used by OQGRG is larger than this value in 6 of the 13 problems. However, the number of GRG calls to find the best solution is larger than that for complete enumeration in only one instance, and the average is 5 versus 66 for complete enumeration. As

with the continuous variable problems, the best solutions are often found in the first GRG call, 7 out of the 13 problems.

Clearly, the number of discrete variables in these problems is too small to infer whether or not this "base-case" OQGRG algorithm will be competitive with alternative MINLP solvers like DICOPT or branch-and-bound [Biegler, et. al., 1997], [Floudas, 1995].

6.6.2 Initial Solution of Relaxed Problem

OQGRG's performance can be significantly enhanced for some problems by sending information on GRG solutions back to OptQuest. The algorithm in this option begins with a call to GRG to solve a relaxed MINLP (with all discrete variables allowed to be continuous). The algorithm can terminate immediately if all discrete variables have discrete values in the GRG solution. Otherwise, the discrete variables are rounded, and the resulting high quality solution is passed to OptQuest to seed the generation of the initial population. Table 6.11 shows the results using this strategy.

Name	Best Known Solution	N0. of Variables	No. of Discrete Vars	Linear Constraint	Nonlinear Constraints	Objective	Best Iteration	Best GRG Call	Total GRG Calls	Total Enum	Locals Found	Function to Best	Total Function Calls	Time to Best	Runtime	GAP
EX12_2_2	1.08	3	1	2	1	1.08	201	2	6	2	2	221	1046	0.3	0.9	-0.01%
EX12_2_1	7.67	5	3	3	2	7.93	0	1	10	8	3	87	1161	0.2	1.9	3.44%
EX12_2_6	-17.00	5	3	4	1	-17.00	0	1	9	8	5	12	1232	0.1	4.3	0.00%
EX9_2_9	2.00	12	3	11	1	2.00	201	2	9	8	4	205	1038	1.9	7.6	0.00%
EX12_2_3	4.58	7	4	5	4	5.58	201	2	18	16	3	366	1280	3.2	3.9	21.84%
EX12_2_3_N	4.58	11	4	9	4	4.58	201	2	14	16	4	314	1218	1.9	2.6	0.00%
EX9_1_9	3.11	17	5	16	1	3.11	201	2	42	32	38	244	1160	9.5	40.6	0.00%
EX12_2_5	31.00	8	6	9	1	31.00	201	2	11	64	5	210	1086	5.6	22.6	0.00%
EX9_1_6	-49.00	20	6	19	1	-49.00	201	2	31	64	29	220	1086	18.0	73.2	0.00%
EX9_1_7	-26.00	23	6	21	1	-26.00	279	6	42	64	37	441	1343	37.4	90.7	0.00%
EX9_1_3	29.20	29	6	27	1	-29.20	201	2	79	64	75	226	1254	32.3	138.0	0.00%
EX12_2_4	-0.94	11	8	4	3	-0.94	202	3	65	256	16	623	3846	12.2	30.8	0.00%
EX12_2_4N	-0.94	11	8	4	3	-0.94	212	5	87	256	5	423	1457	10.7	12.4	0.00%
Averages		**12**	**5**	**10**	**2**		**177**	**2**	**33**	**66**	**17**	**276**	**1401**	**10.2**	**33.1**	

Table 6.11: Solution Statistics for 13 Problems with Discrete Variables, Initial GRG Call Performed on Relaxed Problem

As in the base case, 11 of the 13 problems are solved to the global optimum. EX12_2_4 is also solved beside its reformulated version. EX12_2_3 got worse using this strategy, and one of the smallest problem, EX12_2_1 was not solved to optimality. Interestingly, the average total GRG call is larger in this case, 33 instead of 24. The 2 biggest problems are responsible for most of the increase. The average total function call and runtime are correspondingly larger, too. The iteration, GRG and function calls, and the time to reach the best values are lower, showing that for this limited set of problems the initial relaxed GRG call enhances the performance. The initial call returns the best overall value twice, and

in another 8 cases the GRG call after the Stage 1 global search leads to the optimum. Although the relaxed problem terminated with integer solutions twice, the global optimum was found only for one of them, for EX12_2_6.

6.6.3 Scatter Search Solving the Discrete Sub-problem only

Another way to utilize Scatter Search's efficient handling of the discrete variables is to make it work on a projected version of the original problem. The problem passed to OptQuest has only discrete variables. Also, only constraints where these discrete variables appear linearly are presented to OptQuest. Scatter Search will work on the reduced sub-problem which has only discrete variables and constraints where these discrete variables appear linearly.

Trial solutions recommended from the global search phase are discrete variables that satisfy the linear constraints, thus their number should be no more than that of the complete enumeration. The local solver is called at every iteration. The problem that the local solver solves is the original nonlinear problem with the discrete variables fixed to the level recommended by the trial solution. As a result, the best iteration is the same as the best GRG call. The local solver will return feasible local solutions for the discrete trial solutions where the nonlinear constraints can also be satisfied. Table 6.12 shows the results.

Name	Best Known Solution	N0. of Variables	No. of Discrete Vars	Linear Constraint	Nonlinear Constraints	Objective	Best Iteration	Best GRG Call	Total GRG Calls	Total Enum	Locals Found	Function to Best	Total Function Calls	Time to Best	Runtime	GAP
EX12_2_2	1.08	1	1	0	0	1.08	2	2	2	2	1	15	15	0.3	0.33	-0.01%
EX12_2_1	7.67	3	3	1	0	7.93	3	3	7	8	7	50	128	0.3	0.49	3.44%
EX12_2_6	-17.00	3	3	1	0	-17.00	1	1	6	8	5	10	89	0.3	0.66	0.00%
EX9_2_9	2.00	3	3	0	0	2.00	4	4	8	8	3	112	244	0.3	0.44	0.00%
EX12_2_3_N	4.58	4	4	0	0	5.58	13	13	16	16	16	1025	1360	0.8	1.04	21.84%
EX12_2_3	4.58	4	4	0	0	4.58	13	13	16	16	16	1297	1607	0.8	0.94	0.00%
EX9_1_9	3.11	5	5	0	0	3.11	15	15	32	32	5	487	1086	0.7	1.43	0.00%
EX12_2_5	31.00	6	6	4	0	31.00	22	22	25	64	8	298	331	1	1.48	0.00%
EX9_1_6	-49.00	6	6	0	0	-49.00	38	38	64	64	6	1447	2346	1.8	3.13	0.00%
EX9_1_7	-26.00	6	6	0	0	-26.00	16	16	64	64	14	1179	4782	1.3	4.89	0.00%
EX9_1_3	29.20	6	6	0	0	-29.20	6	6	64	64	14	328	3412	0.5	4.07	0.00%
EX12_2_4	-0.94	8	8	4	0	-0.94	49	49	81	256	81	2083	3305	3.4	5.6	0.00%
EX12_2_4N	-0.94	8	8	4	0	-0.94	49	49	81	256	81	147	243	2.3	3.95	0.00%
Averages		**5**	**5**	**1**	**0**		**18**	**18**	**36**	**66**	**20**	**652**	**1458**	**1.1**	**2.19**	

Table 6.12: Solution Statistics for 13 Problems with Discrete Variables, OptQuest Aware of Only the Discrete Projected Problem

The iteration where the best value is found is low; there is no Stage 1, initial learning phase, in this strategy. As a consequence, the best GRG iteration comes later. The average total GRG calls is only slightly higher: any discrete combination is visited only once. The total GRG calls is still lower than what the total enumeration would be: every trial solution has to satisfy the linear constraints. More local solutions are found since all trial solutions with linearly feasible constraints are evaluated. The function counts are higher in agreement with more GRG calls. However, runtime and the time to reach the best values are significantly lower. This is because OptQuest works only on a significantly

reduced problem. The trade-off is the increased work of the local solver, GRG. It may be possible to reduce GRG's effort by selecting the starting point of the continuous variables to be the values of local solutions whose integer combination is closest to those of the current trial point only.

6.6.4 Reformulation of Discrete Problems

In the 2 reformulated problems, 12_2_3N and 12_2_4N, the discrete variables appear nonlinearly. The versions in the Floudas test set were modified so that these variables appeared linearly. This is required by the DICOPT MINLP solver, which is widely used, especially by chemical engineers. OQGRG allows discrete variables to appear nonlinearly. The two forms of the constraints for problem 12_2_4 are shown below in Table 6.13. Commented lines (starting with an asterisk *) are the versions published in [Floudas et. al., 1999].

The original constraints are derived by taking the logs of the new ones, so that the binary variables *yi*, i = 1,...,8, appears linearly. However, the continuous variables *xi*, i = 1,2,3 appear nonlinearly in the original constraints, but linearly in the new versions. In fact, when OptQuest fixes the binary variables *yi*, the new constraints fix *x1* and *x2* and impose an upper bound on *x3*, so the model with the new constraints is much easier for any NLP solver. That is why all measures of computational effort are much smaller for EX12_2_4N than for EX12_2_4.

```
* MINLP literature problem Berman and Ashrafi, 1993.
* NOTE: The problem has been reformulated so that no
* binary variables appear in nonlinear terms as this
* cannot be handled by GAMS solvers.

VARIABLES
      x1
      x2
      x3
      objval    objective function variable;

FREE VARIABLES    objval;

BINARY VARIABLES
      y1
      y2
      y3
      y4
      y5
      y6
      y7
      y8;

EQUATIONS
      f Objective function
      h1
      h2
      h3
      g1
      g2
      g3
      g4 ;

f   .. objval =e=((-x1)*x2)*x3;
g1  .. -y1-y2-y3 =l= -1;
g2  .. -y4-y5-y6 =l= -1;
g3  .. -y7-y8 =l= -1;
g4  .. 3*y1+y2+2*y3+3*y4+2*y5+y6+3*y7+2*y8 =l= 10;

* original constraints
*h1 .. LOG(0.1)*y1+LOG(0.2)*y2+LOG(0.15)*y3-LOG(1-x1) =e= 0;
*h2 .. LOG(0.05)*y4+LOG(0.2)*y5+LOG(0.15)*y6-LOG(1-x2) =e=0;
*h3 .. LOG(0.02)*y7+LOG(0.06)*y8-LOG(1-x3) =l= 0;

* new constraints
h1 .. x1+(0.1**y1)*(0.2**y2)*(0.15**y3) =e= 1;
h2 .. x2+(0.05**y4)*(0.2**y5)*(.15**y6) =e= 1;
h3 .. x3+ (0.02**y7)*(0.06**y8) =l= 1;
```

```
* Bounds
x1.LO = 0; x1.UP = 1;
x2.LO = 0; x2.UP = 1;
x3.LO = 0; x3.UP = 1;

MODEL test /ALL/;
SOLVE test USING MINLP MINIMIZING objval;
```

Table 6.13: EX12_2_4N Problem; the Original Constraints Commented Out

The model for EX12_2_3_N is shown below in Table 6.14. The commented equations are part of the original formulation, the *yi*, i=1,2,3 are binary variables but the *xyi* variables are constrained only to be positive and upper bounded by 1.0. They are introduced as "surrogates" for the binaries in the nonlinear terms, so that all binary variables will appear linearly.

```
* MINLP literature problem, Yuan et al., 1988.
* NOTE: The problem has been reformulated so that no binary
* variables appear in nonlinear terms as this cannot be
* handled by GAMS solvers.

VARIABLES
         x1
         x2
         x3
*         xy1
*         xy2
*         xy3
*         xy4
         y1
         y2
         y3
         y4
     objval    objective function variable;
```

```
FREE VARIABLES  objval;
BINARY VARIABLE y1;
BINARY VARIABLE y2;
BINARY VARIABLE y3;
BINARY VARIABLE y4;

EQUATIONS
         g1
         g2
         g3
         g4
         g5
         g6
         g7
         g8
         g9
*         g10
*         g11
*         g12
*         g13
         f Objective function;

f  .. objval =e=POWER(y1-1,2)+POWER(y2-2,2)+POWER(y3-1,2)
-(LOG(y4+1))+POWER(x1-1,2)+POWER(x2-2,2)+POWER(x3-3,2);

* original objective, with binaries replaced by
continuous variables
*f  .. objval =e=POWER(xy1-1,2)+POWER(xy2-2,2)+POWER(xy3-
1,2)-(LOG(xy4+1))+POWER(x1-1,2)+POWER(x2-2,2)+POWER(x3-3,2);

g1  .. y1+y2+y3+x1+x2+x3 =l= 5;
g2  .. POWER(y3,2)+POWER(x1,2)+POWER(x2,2)+POWER(x3,2)=l=
      5.5;
*g2  .. POWER(xy3,2)+POWER(x1,2)+POWER(x2,2)+POWER(x3,2)=l=
      5.5;
g3  .. y1+x1 =l= 1.2;
g4  .. y2+x2 =l= 1.8;
g5  .. y3+x3 =l= 2.5;
g6  .. y4+x1 =l= 1.2;

* constraints where binary variables appear nonlinearly
g7  .. POWER(y2,2)+POWER(x2,2) =l= 1.64;
g8  .. POWER(y3,2)+POWER(x3,2) =l= 4.25;
g9  .. POWER(y2,2)+POWER(x3,2) =l= 4.64;
*g7  .. POWER(xy2,2)+POWER(x2,2) =l= 1.64;
*g8  .. POWER(xy3,2)+POWER(x3,2) =l= 4.25;
*g9  .. POWER(xy2,2)+POWER(x3,2) =l= 4.64;
*g10  .. xy1-y1 =e= 0;
*g11  .. xy2-y2 =e= 0;
*g12  .. xy3-y3 =e= 0;
```

```
*g13  .. xy4-y4 =e= 0;

* Bounds
x1.LO = 0; x1.UP = 10;
x2.LO = 0; x2.UP = 10;
x3.LO = 0; x3.UP = 10;
*xy1.LO = 0; xy1.UP = 1;
*xy2.LO = 0; xy2.UP = 1;
*xy3.LO = 0; xy3.UP = 1;
*xy4.LO = 0; xy4.UP = 1;

MODEL test /ALL/;
SOLVE test USING MINLP MINIMIZING objval;
```

Table 6.14: EX12_2_3_N Problem; the Original Constraints Commented Out

6.7 The Lennard-Jones and Morse Energy Minimization Problems

The Floudas set of test problems includes two GAMS models that minimize the potential energy of a cluster of N particles, using two different potential energy functions. The model EX8_6_1, using the Lennard-Jones potential function with N = 5 is shown in Table 6.15. The problem can be reformulated into an unconstrained model. The original formulation is commented out.

```
*-----------------------------------*
* Scaled Lennard-Jones Test Problem *
*-----------------------------------*
Sets
    i    number of particles    /1*5/
    iter number of local mins   /1*1/
    alias(i,j);

Scalars
    bnd absolute value of bound /5.0/;

Parameters
    b(i,j)  on-off for interactions;

loop(i, loop(j,
    b(i,j) = 1 $ (ord(i) lt ord(j))));

Variables
    x(i)       x coordinates
    y(i)       y coordinates
    z(i)       z coordinates
*    r2inv(i,j) inverse squared interparticle distance
    f          potential energy;

loop(i,
    x.lo(i) = -bnd $ (ord(i) ge 2);
    y.lo(i) = -bnd $ (ord(i) ge 3);
    z.lo(i) = -bnd $ (ord(i) ge 4);
```

```
    x.up(i) =  bnd $ (ord(i) ge 2);
    y.up(i) =  bnd $ (ord(i) ge 3);
    z.up(i) =  bnd $ (ord(i) ge 4));

x.lo('1') = 0; x.up('1') = 0;
y.lo('1') = 0; y.up('1') = 0;
z.lo('1') = 0; z.up('1') = 0;
y.lo('2') = 0; y.up('2') = 0;
z.lo('2') = 0; z.up('2') = 0;
z.lo('3') = 0; z.up('3') = 0;

Equations
    obj    objective function;
*    d(i,j) distance equations;

obj ..
    f =e= sum((i,j) $ b(i,j),
*          power(r2inv(i,j),6) - 2*power(r2inv(i,j),3));
           power((1 / (power(x(i)-x(j),2)
                   + power(y(i)-y(j),2)
                   + power(z(i)-z(j),2))),6)
         - 2*power((1 / (power(x(i)-x(j),2)
                   + power(y(i)-y(j),2)
                   + power(z(i)-z(j),2))),3)) ;

*d(i,j) $ b(i,j) ..
*    r2inv(i,j) =e= 1 / (power(x(i)-x(j),2)
*                     + power(y(i)-y(j),2)
*                     + power(z(i)-z(j),2));

Model
*    problem /obj, d/;
    problem /obj/;

solve problem using nlp minimizing f;
```

Table 6.15: The Lennard-Jones Energy Minimization Problem, EX8_6_1

The decision variables are the x, y, and z components of each particle. The objective is the summed difference between the sixth and third powers of the reciprocal of the squared Euclidean distance between each distinct pair of particles, where the sixth power term arises from a strong short-range repulsive force and the other term from a longer-range attractive force. Particle 1 is located at the origin, and three position components of particles 2 and 3 are fixed, so this family of problems has N-6 variables and N(N-1) nonlinear constraints.

The second set of problems, EX8_6_2, is as above but uses the Morse potential, given in Table 6.16.

```
f =e= sum((i,j) $ b(i,j),
      power(1-exp(3*(1- (power(x(i)-x(j),2) +
      power(y(i)-y(j),2) +
      power(z(i)-z(j),2))**0.5)), 2) - 1);
```

Table 6.16: The Objective Function of the Morse Energy Minimization Problem, EX8_6_2

According to [Floudas, 1999, pp. 186-194], these problems have a large number of local minima, and this number increases rapidly with problem size. Thus these problems can form a rigorous test for global optimization algorithms. Results of applying OQGRG to these two problem classes using 200 stage one and 1000 total iterations for several values of N are shown in Tables 6.17 and 6.18.

N	Variables	Objective	Best Iteration	Best GRG Call	Total GRG Calls	GRG Ratio	Locals Found	Function to Best	Total Functions	Function Ratio	Time to Best	Runtime	Time Ratio	Gap
5	9	-9.10	201	1	5	0.20	3	312	2676	0.12	0.3	2.5	0.13	0.0%
10	24	-28.42	349	5	14	0.36	14	2563	7615	0.34	6.0	17.6	0.34	0.0%
15	39	-52.32	407	11	56	0.20	56	7431	34265	0.22	33.5	152.1	0.22	0.0%
20	54	-75.59	302	6	38	0.16	38	6441	29584	0.22	53.0	238.4	0.22	2.1%
25	69	-99.37	999	45	45	1.00	45	41678	41679	1.00	530.1	530.7	1.00	2.9%
30	84	-125.85	682	36	66	0.55	66	47232	83730	0.56	853.7	1510.0	0.57	1.9%
avg	**47**		**490**	**17**	**37**	**0.41**	**37**	**17610**	**33258**	**0.41**	**246**	**408.55**	**0.41**	**1.1%**

Table 6.17: Minimizing the Lennard-Jones Potential Function

N	Variables	Objective	Best Iteration	Best GRG Call	Total GRG Calls	GRG Ratio	Locals Found	Function to Best	Total Functions	Function Ratio	Time to Best	Runtime	Time Ratio	Gap
5	9	-9.30	201	1	9	0.11	6	342	2174	0.16	0.33	1.76	0.19	0.0%
10	24	-31.89	284	2	3	0.67	3	922	2067	0.45	1.65	3.62	0.46	0.0%
15	39	-63.16	201	1	13	0.08	13	533	6035	0.09	2.14	19.28	0.11	0.0%
20	54	-97.42	201	1	38	0.03	38	673	17747	0.04	4.72	89.04	0.05	0.0%
25	69	-136.07	628	11	12	0.92	12	6400	7314	0.88	52.3	61.02	0.86	0.0%
30	84	-177.58	886	34	46	0.74	46	22149	29234	0.76	252	330.7	0.76	0.0%
40	114	-267.62	560	13	15	0.87	15	11195	13024	0.86	234	284.3	0.82	0.3%
50	144	-366.62	750	24	41	0.59	41	23140	38490	0.60	805	1313	0.61	0.0%
avg	**84**		**538**	**14**	**28**	**0.5**	**28**	**10682**	**18641**	**0.5**	**225.0**	**349.6**	**0.5**	**0.0**

Table 6.18: Minimizing the Morse Potential Function

OQGRG finds the Morse potential easier to minimize, and solves 7 of the 8 instances to essentially zero gaps and the N=45 case to a 0.3% gap. The 3 smallest instances of the Lennard-Jones problems are also solved to near-zero gaps, but the next three have gaps of 2.1%, 2.9%, and 1.9% respectively. This is probably due to the large number of local solutions for these problems.

The computational effort needed to achieve these good results is quite modest. As before, the ratio columns are the effort to find the best solution divided by total effort, and these ratios are close to 0.5 for the Morse potential, and occasionally above 0.4 for the Lennard-Jones. The number of local minima found increases rapidly with N, and for N=30 it is typically above 40. This number is usually equal to the number of GRG calls, so GRG almost always finds a different local solution at each start.

Results of solving the Lennard-Jones problems with gaps larger than 1% using 200 stage one and 3000 total iterations are shown in table 6.19 below.

N	Variables	Objective	Best Iteration	Best GRG Call	Total GRG Calls	GRG Ratio	Locals Found	Function to Best	Total Functions	Function Ratio	Time to Best	Runtime	Time Ratio	Gap
EX8_6_1RN:														
20	54	-75.59	299	4	274	0.01	274	3389	157146	0.02	28.8	1247.0	0.02	2.06%
25	69	-101.82	869	34	234	0.15	234	33231	204556	0.16	423.4	2581.6	0.16	0.54%
30	84	-127.23	2778	236	254	0.93	254	260619	279494	0.93	4709.7	5051.1	0.93	0.82%
EX8_6_2RN:														
40	114	-267.62	561	14	16	0.88	16	12256	16084	0.76	257.8	409.8	0.63	0.29%

Table 6.19: Solving 4 Problems with 3000 Iterations

The EX8_6_1R25 and EX8_6_1R30 problems are solved to a less-than 1% gap. The overall effort for these 2 problems increases almost eight-fold measured in GRG calls. Interestingly, the best iteration is less than 1000 and there is a better solution than for the base case. This is possible because some aspects of the OptQuest solution strategy depend on the iteration limit, so the two runs use a different sequence of trial points, even in their first 1000 iterations. The search is more aggressive when there are only 1000 iterations allowed, and this aggressiveness leads to a worse final solution in the shorter run.

There are no improvements in solving EX8_6_1R20 and EX8_6_2R40, despite the large increase in GRG calls for EX8_6_1R20. EX8_6_2R40 has only one more GRG calls for 3000 total iterations. This suggests that a relaxation in the OQGRG filtering parameters might help. The parameter *waitcycle* is changed to 10, *relax-factor* is 0.3, and the *basin-factor* is 0.5. Table 6.20 shows the result.

N	Variables	Objective	Best Iteration	Best GRG Call	Total GRG Calls	GRG Ratio	Locals Found	Function to Best	Total Functions	Function Ratio	Time to Best	Runtime	Time Ratio	Gap
EX8_6_2R40:														
40	114	-268.39	584	39	105	0.37	105	34574	81113	0.43	707.9	1650	0.43	0.00%

Table 6.20: Solving with Relaxed OQGRG Filtering Parameters

Total GRG calls have gone up to 105, and there are more function calls as well. The global optimum has been found.

6.8 Changing Algorithm Options to Solve Difficult Problems

In this section some of the problems that are not solved in the base case, or with changes mentioned subsequently, are discussed. One of the typical reasons for not getting to a local optimum is not performing enough iterations. If we increase the total iteration limit to 3000 or 5000 from the default 1000, some of the problems will reach better values than the base case, sometimes even the best known value. This has already been shown for 3000 iterations for the Lennard-Jones problems in Table 6.19. For additional problems, see Table 6.21 below.

Problem	Variables	Objective	Best Iteration	Best GRG Call	Total GRG Calls	GRG Ratio	Locals Found	Function to Best	Total Functions	Function Ratio	Time to Best	Runtime	Time Ratio	Gap
EX2_1_1	6	-17.00	893	2	14	0.14	12	896	5055	0.18	0.2	1.3	0.17	0.00%
EX2_1_7_4	21	-754.75	2543	46	60	0.77	22	4171	7043	0.59	57.8	88.2	0.66	0.00%
EX2_1_7_5	21	-4150.40	1965	19	31	0.61	16	2179	5465	0.40	38.1	78.5	0.49	0.00%
EX2_1_8	25	15639.00	887	23	29	0.79	10	1483	5742	0.26	6.0	39.1	0.15	0.00%

Table 6.21: Solving Some Problems with 5000 Total Iterations

The problems of EX2_1_x have quadratic objectives and linear constraints. The global optimum can be found among the vertices of these problems. This insight suggests that the global search should be directed towards the edge of the domain: increasing the parameter of the boundary strategy of OptQuest to 0.8 indeed improves EX2_1_6, as shown in Table 6.22.

Problem	Variables	Objective	Best Iteration	Best GRG Call	Total GRG Calls	GRG Ratio	Locals Found	Function to Best	Total Functions	Function Ratio	Time to Best	Runtime	Time Ratio	Gap
EX2_1_6	11	-39.00	323	6	20	0.3	6	408	1203	0.34	1.37	1.87	0.73	0.00%

Table 6.22: Solving EX2_1_6 with Boundary Strategy Value = 0.8

The chemical reactor network synthesis problems (EX8_3_x) are the largest ones in the test set with more than 100 variables and several dozen constraints. They have many local optima which make them hard to solve to global optimality. Table 6.23 shows results with the iteration number set to 5000 and the GRG feasibility and stopping condition tolerances set to 10^{-6} for the problems that did not solve in the base case.

Problem	Variables	Objective	Best Iteration	Best GRG Call	Total GRG Calls	GRG Ratio	Locals Found	Function to Best	Total Functions	Function Ratio	Time to Best	Runtime	Time Ratio	Gap
EX8_3_14	111	2.41	952	81	413	0.20	30	252113	1177894	0.21	486	2648	0.18	3.4%
EX8_3_2	111	0.41	201	1	20	0.05	20	2031	17913	0.11	27	533	0.05	0.0%
EX8_3_5	111	0.07	1151	51	94	0.54	91	70987	135419	0.52	189	674	0.28	0.6%
EX8_3_1	116	0.78	210	4	40	0.10	34	7527	67564	0.11	45	571	0.08	4.8%
EX8_3_11	116	0.76	209	5	30	0.17	27	17224	1884537	0.01	52	2794	0.02	4.9%
EX8_3_13	116	51.49	837	16	46	0.35	32	97196	332750	0.29	184	871	0.21	-19.5%
EX8_3_12_C	121	1.00	4107	215	288	0.75	204	533525	918272	0.58	1095	1655	0.66	0.1%
EX8_3_7	127	1.23	375	11	18	0.61	12	22750	30316	0.75	73	552	0.13	0.0%
EX8_3_8	127	3.26	1172	30	158	0.19	145	157408	455296	0.35	421	1366	0.31	0.0%

Table 6.23: EX8_3_x problems with 5000 Total Iterations and 10^{-6} Tolerance Values

Better than the best know solution was found with these parameters for EX8_3_13. Five more problems had solution gaps less than 1%, practically having these problems solved to global optimality. For the remaining 3 problems the gap shrank below 5%.

7. Summary and Opportunities

Previous chapters have introduced Global Optimization and then discussed the OQGRG algorithm and its 2 major components, OptQuest, an implementation of the Scatter Search meta-heuristic, and GRG, an implementation of a reduced gradient method. The power of the OQGRG algorithm has been demonstrated by solving a test set consisting of 163 problems. Over 80% of the problems were successfully solved by the recommended base case option of OQGRG. Using all available options, OQGRG solved 155 (97.5%) problems out of 159 (the original 163 minus the 4 problems for which GRG has never found any feasible solutions) within 1% of the best known solutions. The remaining 4 problems were all within 5% of the best known solutions. Several ways to change the algorithm's options were shown to illustrate how these options can be utilized to make the algorithm perform better on some problems.

There are ample opportunities for future work related to OQGRG. The algorithm can be further modified and adapted to use special properties of some problem classes. An example of the latter is using the fact that concave quadratic programs have optimal vertex solutions by directing the trial points toward the boundary of the feasible region. Performance of the Scatter Search, the global search phase, could be improved by communicating more GRG solutions back to OptQuest. The option of placing a GRG solution into OptQuest's initial

population has been investigated, but sending GRG solutions back to OptQuest in Stage 2 of the OQGRG algorithm needs further attention.

Discrete problems pose some additional challenges when the number of discrete variables grows. Starting points for GRG, the local solver, can be selected to reduce the effort it takes to find local optima, by starting from the previously found local solution whose integer variables are closest to the current ones. MINLP problems with more discrete variables must be obtained and solved to evaluate this and other possible improvements.

NLP algorithms can fail by being unable to find a feasible point in cases where the problem instance is feasible. With GRG algorithms, this usually happens when Phase 1 terminates at a local optimum of the Phase 1 objective. OQGRG can be applied to such problems, if they are reformulated by dropping the true objective, adding deviation variables into all constraints, and minimizing the sum of these deviation variables. This approach could greatly improve the ability of existing NLP solvers to diagnose infeasibility, but this must be shown empirically.

Solving additional test problems will certainly lead to new ideas on how to improve the algorithm. Many lessons were learned in the course of solving the described ~160 test problems. New test problems will most certainly bring new challenges and new ideas to the algorithm.

The field of Global Optimization is changing rapidly, new algorithms and approaches appear regularly. Comparing the performance of OQGRG to the performance of Frontline System's MLSL, Janos Pintér's LGO, and I. Grossman's DICOPT++ could be of great interest – once all the solvers are interfaced to the same platform so meaningful comparisons can be made. The online "MINLP World" being developed by GAMS development company (see www.gamsworld.org/minlp) will provide more test problems and make such comparisons much easier for MINLPs. A related "Global World" initiative will provide other global optimization examples, and it will make it possible to compare the results of OQGRG with results obtained using other solvers. Such comparisons are needed in order to better understand the relative strengths and weaknesses of all existing approaches to Global Optimization, and to assess the contribution of the OQGRG.

APPENDIX A The oqalgpar.stg settings file

```
*
********** OQGRG Alg parameters
*
* use linear constraints in setting up Optquest:
USE_LINEAR_CONSTRAINTS:             1
* initial GRG call 0-none; 1-with fixed integers; 2-relaxed:
START_WITH_GRG:                     0
* use rnd starting pts for initial GRG; effective only if
* START_WITH_GRG>0 and initial point is all zeros:
RANDOM_INITIAL_PTS:                 0
* use only Optquest iterations (0-OQGRG, 1-no GRG calls are made):
OPTQUEST_ONLY:                      0
* OptQuest only knows about constraints with discrete vars:
DISCRETES_ONLY_OPTQUEST:            1
* total number of OQGRG iterations:
OQGRG_TOTALITER:                    1000
* number of cycles for trial solutions from population before
* algorithm start; If 0, use fixed number of PURE_OQ_ITR:
INIT_CYCLE:                         1
* number of fixed initial optquest iterations:
PURE_OQ_ITR:                        20
* fixed bound (NOT multiplying factor for bound calculation!):
BNDMULT:                            100
* starting value for Lagrange multipliers:
STARTING_MULTIPLIER:                1000
* number of itns b/f relaxing the threshold in the Merit Filter:
WAITCYCLE:                          20
* factor to relax the Merit Filter threshold:
RELAXFACTOR:                        0.2
* penalty factor used to calculate merit function value:
PENALTY_FACTOR:                     5
* factor used to adjust maxdist for Distance Filter:
BASIN_FACTOR:                       0.75
* factor used to adjust maxdist for Distance Filter for infeasible
* points:
INF_NBRHD_FACTOR:                   0.20
* use normalized distance for distance calculation (yes/no):
NORM_DIST:                          0
* oqgrg tolerance limit:
TOLERANCE:                          0.0001
* return local solutions to optquest (yes/no):
LOCAL_RETURN:                       0
* use only discretized variables in optquest search (yes/no):
OQ_ALL_DISCRETE:                    0
* discretization step used:
OQ_DISCRETE_STEP:                   0.1
```

```
* apply Merit Filtering (yes/no):
USE_MERIT_F_VALUE_LOGIC:            1
* apply Distance Filtering (yes/no):
USE_X_DISTANCE_LOGIC:               1
* penalty fcn used in choosing best initial trial solution:
* (0-OptQuest's, 1-exact penalty):
USE_OQGRG_PENALTY:                  1
* apply the Merit Filtering individually to all integer
combination as separate problems (yes/no):
APPLY_MERIT_ON_INT_COMB:            1
* apply the Distance Filtering individually to all integer
combination as separate problems (yes/no):
APPLY_BASIN_ON_INT_COMB:            1
* maximum runtime:
*MAXTIME_OQGRG:                     60
*
*
********** OPTQUEST parameters
*
* optquest population size:
OQGRG_POPSIZE:                      15
* optquest objective value accuracy in digits:
OQGRG_OBJ_PRECISION:                4
* optquest variable value accuracy in digits:
OQGRG_VAR_PRECISION:                4
* OQ search strategy type: agressive, boundary, crossover,
permutation, or parallel search
SEARCH_TYPE:                        boundary
* Parameter for the search strategy types
SEARCH_PARAMETER:                   0.5
* Optquest database feature used:
OQ_DATABASE:                        1
* setting the random seed for OptQuest:
RANDOMSEED:                         670729
*
*
********** GRG parameters
*
* epfeas
GRG_EPFEAS:                         0.0001
* epstop
GRG_EPSTOP:                         0.0001
* nstop
GRG_NSTOP:                          6
* use phase0
GRG_USE_PHASE0:                     1
* apply jacobian scaling at initial point
GRG_SCALING_ON:                     0
* periodic jacobian re-scaling every GRG_ISCLAG linesearches
(0=OFF)
GRG_ISCLAG:                         0
```

```
* initial printing
GRG_INPRNT:                          1
* final printing
GRG_OTPRNT:                          1
* printlevel set
GRG_PRINTLEVEL:                      1
*
*
********** PRINT control
*
* turning on/off printing trial solutions into a file:
PRINT_TRIAL_POINTS:                  1
* turning on/off printing populations into a file:
PRINT_POPULATION:                    1
* setting the frequency of itn print to standard output:
PRINT_ITERATION:                     100
* turning on different level of debug output:
OQGRG_DEBUG:                         2
```

APPENDIX B The EX8_1_5.itn output file

```
  EX8_1_5

  nvars_in  =      3  nfun_in    =      1  nobj_in  =      1
  TotalIter =   1000  InitOQItrs =    200  PopSize  =     15
  UseLinCnst=      1  maximize   =      0  Database =      1
  WaitCycle =     20  ObjPrec    =      4  VarPrec  =      4
  StartW/GRG=      0  OQ Only    =      0
  BndMult   = 100.00  RelaxFactor=   0.20  PnltyFact=   5.00
  BsnFactor =   0.75  InfBsnFactr=   0.20  SearchPrm=   0.50
  SrchType  =        BOUNDARY


iter,,obj,,termin,ninf,itns,ngcomps,,initial points,,final points
154,,0.000000,,KTC,0,0,2,,0.000000,0.000000,,0.000000,0.000000
200,,-1.031628,,KTC,0,4,20,,-0.21919,-0.414566,,0.089842,-0.712656
227,,-1.031628,,KTC,0,5,25,,-0.11303,-0.279056,,0.089842,-0.712656
264,,-0.215464,,KTC,0,3,15,,-1.648610,0.825989,,-1.703604,0.796083
267,,-1.031628,,KTC,0,4,20,,-0.360287,0.662640,,-0.089842,0.712656
324,,-1.031628,,KTC,0,5,24,,0.207671,0.865897,,-0.089842,0.712656
359,,-1.031628,,KTC,0,5,23,,0.209152,0.775845,,-0.089842,0.712656
362,,-1.031628,,KTC,0,3,18,,0.210632,0.685793,,-0.089842,0.712656
367,,-1.031628,,KTC,0,3,17,,0.176127,0.670283,,-0.089842,0.712658
549,,-1.031628,,KTC,0,4,20,,-0.376868,0.694788,,-0.089842,0.712656


#1 lowest local optimum
  g:    -1.0316,    -1.0316, x:     0.0898,    -0.7127,
Solution first found at iteration 200.
Solution found 2 times.

#2 lowest local optimum
  g:    -1.0316,    -1.0316, x:    -0.0898,     0.7127,
Solution first found at iteration 267.
Solution found 6 times.

#3 lowest local optimum
  g:    -0.2155,    -0.2155, x:    -1.7036,     0.7961,
Solution first found at iteration 264.
Solution found 1 times.

#4 lowest local optimum
  g:     0.0000,     0.0000, x:     0.0000,     0.0000,
Solution first found at iteration 154.
Solution found 1 times.
```

APPENDIX C The EX8_1_5.oqg output file (shortened)

```
  EX8_1_5

  nvars_in  =      3  nfun_in    =      1  nobj_in  =      1
  TotalIter =   1000  InitOQItrs =    200  PopSize  =     15
  UseLinCnst=      1  maximize   =      0  Database =      1
  WaitCycle =     20  ObjPrec    =      4  VarPrec  =      4
  StartW/GRG=      0  OQ Only    =      0
  BndMult   = 100.00  RelaxFactor=   0.20  PnltyFact=   5.00
  BsnFactor =   0.75  InfBsnFactr=   0.20  SearchPrm=   0.50
  SrchType  =        BOUNDARY


  Itn      Obj     Mer    PenVal    Threshold  Dst    GRGObj   Trm
Sinf
     1 +0.000e+000
     2 +3.524e+005
     3 +3.524e+005
     4 +1.101e+005
     5 +1.261e+005
     6 +3.930e+004
     7 +2.576e+004
     8 +2.041e+005
     9 +2.712e+005
    10 +3.488e+003
    11 +3.188e+001
    12 +6.611e+002
    13 +2.070e+004
    14 +6.421e+003
    15 +8.915e-003
    16 +8.915e-003
    17 +4.764e+004
    18 +1.567e+002
    19 +1.567e+002
    20 +5.137e+004
    21 +1.064e+003
    22 +1.064e+003
    23 +3.522e+005
    24 +3.470e-001

  Itn      Obj     Mer    PenVal    Threshold  Dst    GRGObj   Trm
Sinf
    25 +3.470e-001
    26 +3.522e+005
```

```
   146 +2.785e+002
   147 +2.912e+005
   148 +3.174e+005
   149 +9.372e+001

  Itn      Obj     Mer    PenVal    Threshold  Dst    GRGObj   Trm
Sinf
   150 +1.658e+002
   151 +2.474e+003
   152 +3.034e+005
   153 +7.029e+002
   154 +7.029e+002

Trial solution cycle 1 end
   154 +0.000e+000   Best Pt from Initial Iteration loop
   154 +0.000e+000 ACC +1.000e+030 -1.000e+030 ACC +0.000e+000 KTC
+0.000e+000
   155 +2.980e+002 REJ +2.980e+002 +0.000e+000 ACC
   156 +2.596e+002 REJ +2.596e+002 +0.000e+000 ACC
```



```
   196 +2.901e+002 REJ +2.901e+002 +4.400e-001 ACC
   197 +1.376e+000 REJ +1.376e+000 +4.400e-001 ACC
   198 +2.463e+000 REJ +2.463e+000 +4.400e-001 ACC
   199 +1.552e+004 REJ +1.552e+004 +4.400e-001 ACC

  Itn      Obj     Mer    PenVal    Threshold  Dst    GRGObj   Trm
Sinf
   200 -2.911e-001 ACC -2.911e-001 +4.400e-001 ACC -1.032e+000 KTC
+0.000e+000

  Itn      Obj     Mer    PenVal    Threshold  Dst    GRGObj   Trm
Sinf
   201 +1.976e-001 REJ +1.976e-001 -2.911e-001 ACC
   202 +3.524e+005 REJ +3.524e+005 -2.911e-001 ACC
```



```
   223 +1.562e+004 REJ +1.562e+004 -3.286e-002 ACC
   224 +2.475e-001 REJ +2.475e-001 -3.286e-002 ACC

  Itn      Obj     Mer    PenVal    Threshold  Dst    GRGObj   Trm
Sinf
   225 +2.475e-001 REJ +2.475e-001 -3.286e-002 ACC
   226 +8.885e+000 REJ +8.885e+000 -3.286e-002 ACC
   227 -2.049e-001 ACC -2.049e-001 -3.286e-002 ACC -1.032e+000 KTC
+0.000e+000
   228 -2.049e-001 REJ -2.049e-001 -2.049e-001 ACC
   229 +1.511e+001 REJ +1.511e+001 -2.049e-001 ACC
   230 +2.693e+003 REJ +2.693e+003 -2.049e-001 ACC
```

```
   261 +1.562e+004 REJ +1.562e+004 +3.606e-002 ACC
   262 +3.150e+005 REJ +3.150e+005 +3.606e-002 ACC
   263 +1.023e+004 REJ +1.023e+004 +3.606e-002 ACC
   264 -1.776e-001 ACC -1.776e-001 +3.606e-002 ACC -2.155e-001 KTC
+0.000e+000
   265 -1.432e-001 REJ -1.432e-001 -1.776e-001 ACC
   266 +3.003e+003 REJ +3.003e+003 -1.776e-001 ACC
   267 -7.393e-001 ACC -7.393e-001 -1.776e-001 ACC -1.032e+000 KTC
+0.000e+000
   268 +1.430e+000 REJ +1.430e+000 -7.393e-001 ACC
   269 +1.476e+004 REJ +1.476e+004 -7.393e-001 ACC
   270 +2.173e+000 REJ +2.173e+000 -7.393e-001 ACC
   …
   320 +7.432e+000 REJ +7.432e+000 -1.132e-001 ACC
   321 +1.543e+004 REJ +1.543e+004 -1.132e-001 ACC
   322 +2.359e+003 REJ +2.359e+003 -1.132e-001 ACC
   323 +4.544e+000 REJ +4.544e+000 -1.132e-001 ACC
   324 -4.020e-001 ACC -4.020e-001 -1.132e-001 ACC -1.032e+000 KTC
+0.000e+000

  Itn      Obj      Mer    PenVal    Threshold  Dst    GRGObj   Trm
Sinf
   325 +2.129e+001 REJ +2.129e+001 -4.020e-001 ACC
   326 -9.195e-001 ACC -9.195e-001 -4.020e-001 REJ
```

```
   358 +1.355e+004 REJ +1.355e+004 -6.136e-001 ACC
   359 -6.252e-001 ACC -6.252e-001 -6.136e-001 ACC -1.032e+000 KTC
+0.000e+000
   360 +4.844e-002 REJ +4.844e-002 -6.252e-001 ACC
   361 -1.154e-001 REJ -1.154e-001 -6.252e-001 ACC
   362 -6.787e-001 ACC -6.787e-001 -6.252e-001 ACC -1.032e+000 KTC
+0.000e+000
   363 -5.960e-001 REJ -5.960e-001 -6.787e-001 ACC
   364 +4.003e+000 REJ +4.003e+000 -6.787e-001 ACC
   365 -3.726e-001 REJ -3.726e-001 -6.787e-001 ACC
   366 -3.221e-002 REJ -3.221e-002 -6.787e-001 ACC
   367 -7.496e-001 ACC -7.496e-001 -6.787e-001 ACC -1.032e+000 KTC
+0.000e+000
   368 -5.469e-001 REJ -5.469e-001 -7.496e-001 ACC
   369 +4.459e+002 REJ +4.459e+002 -7.496e-001 ACC
```

```
   545 -8.589e-001 REJ -8.589e-001 -1.021e+000 REJ
   546 -7.893e-001 REJ -7.893e-001 -1.021e+000 REJ
   547 -9.087e-001 REJ -9.087e-001 -1.021e+000 REJ
   548 -6.510e-001 REJ -6.510e-001 -1.021e+000 ACC
   549 -7.339e-001 ACC -7.339e-001 -6.169e-001 ACC -1.032e+000 KTC
+0.000e+000
```

```
  Itn      Obj     Mer    PenVal    Threshold  Dst    GRGObj   Trm
Sinf
   550 -5.138e-001 REJ -5.138e-001 -7.339e-001 ACC
   551 -9.557e-001 ACC -9.557e-001 -7.339e-001 REJ
```



```
   994 +6.027e+003 REJ +6.027e+003 -1.032e+000 ACC
   995 -1.031e+000 REJ -1.031e+000 -1.032e+000 REJ
   996 -1.031e+000 REJ -1.031e+000 -1.032e+000 REJ
   997 +3.522e+005 REJ +3.522e+005 -1.032e+000 ACC
   998 +7.405e+003 REJ +7.405e+003 -1.032e+000 ACC
   999 +3.323e+005 REJ +3.323e+005 -6.253e-001 ACC

  Itn      Obj     Mer    PenVal    Threshold  Dst    GRGObj   Trm
Sinf
  1000 +4.564e+003 REJ +4.564e+003 -6.253e-001 ACC



Merit only rejected                499 times;
Basin only rejected                 57 times;
Both  Filter rejected              281 times;
GRG was called                      10 times;
GRG returned infeasible solution     0 times;


#1 lowest local optimum
  g:    -1.0316,    -1.0316, x:     0.0898,    -0.7127,
Solution first found at iteration 200.
Solution found 2 times.

#2 lowest local optimum
  g:    -1.0316,    -1.0316, x:    -0.0898,     0.7127,
Solution first found at iteration 267.
Solution found 6 times.

#3 lowest local optimum
  g:    -0.2155,    -0.2155, x:    -1.7036,     0.7961,
Solution first found at iteration 264.
Solution found 1 times.

#4 lowest local optimum
  g:     0.0000,     0.0000, x:     0.0000,     0.0000,
Solution first found at iteration 154.
Solution found 1 times.
```

APPENDIX D The EX8_1_5.LST output file

```
…
GAMS Rev 116  Windows NT/95/98                    07/14/01
18:19:12  PAGE      5
G e n e r a l  A l g e b r a i c  M o d e l i n g  S y s t e m


               S O L V E      S U M M A R Y

     MODEL   test                OBJECTIVE  objval
     TYPE    NLP                 DIRECTION  MINIMIZE
     SOLVER  OQGRG               FROM LINE  31

**** SOLVER STATUS     1 NORMAL COMPLETION
**** MODEL STATUS      2 LOCALLY OPTIMAL
**** OBJECTIVE VALUE               -1.0316

 RESOURCE USAGE, LIMIT          1.480     1000.000
 ITERATION COUNT, LIMIT      1184        10000
 EVALUATION ERRORS              0            0


  OptQuest/Grg  Release of February 10, 2001
  ------------------------------------------

                       LOWER     LEVEL     UPPER    MARGINAL

---- EQU f               .       -1.032      .         .

  f  Objective function

                       LOWER     LEVEL     UPPER    MARGINAL

---- VAR x            -10.000     0.090    10.000      .
---- VAR y            -10.000    -0.713    10.000      .
---- VAR objval         -INF     -1.032     +INF       .

  x
  y
  objval  objective function variable


**** REPORT SUMMARY :        0     NONOPT
                             0 INFEASIBLE
                             0  UNBOUNDED
                             0     ERRORS
```

```
EXECUTION TIME       =        0.000 SECONDS    0.7 Mb      WIN194-
116


USER: Leon Lasdon
G000927:1441AV-WIN
      University of Texas at Austin, MSIS
DC1504

**** FILE SUMMARY

INPUT      C:\MYRUNS\OQGRG\EX8_1_5.GMS
OUTPUT     C:\MYRUNS\OQGRG\EX8_1_5.LST
```

APPENDIX E Test Set Problem Statistics

Name	Variables	Discretes	lin. constr.	nl constr.	best known obj.
EX2_1_1	6	0	1	0	-17
EX2_1_10	21	0	10	0	49318.018
EX2_1_2	7	0	2	0	-213
EX2_1_3	14	0	9	0	-15
EX2_1_4	7	0	5	0	-11
EX2_1_5	11	0	11	0	-268.0146
EX2_1_6	11	0	5	0	-39
EX2_1_7_1	21	0	10	0	-394.7506
EX2_1_7_2	21	0	10	0	-884.75058
EX2_1_7_3	21	0	10	0	-8695.01193
EX2_1_7_4	21	0	10	0	-754.75062
EX2_1_7_5	21	0	10	0	-4150.4101
EX2_1_8	25	0	10	0	15639
EX2_1_9	11	0	1	0	-0.375
EX3_1_1	9	0	3	3	7049.248
EX3_1_2	6	0	0	6	-30665.5254
EX3_1_3	7	0	4	2	-310
EX3_1_4	4	0	2	1	-4
EX4_1_1	2	0	0	0	-7.4873
EX4_1_2	2	0	0	0	-663.5001
EX4_1_3	2	0	0	0	-443.6717
EX4_1_4	2	0	0	0	0
EX4_1_5	3	0	0	0	0
EX4_1_6	2	0	0	0	7
EX4_1_7	2	0	0	0	-7.5
EX4_1_8	3	0	0	1	-16.7392
EX4_1_9	3	0	0	2	-5.50796
EX5_2_4	8	0	3	3	450
EX5_2_5	33	0	8	11	3500.0006
EX5_3_2	23	0	7	9	1.8642
EX5_3_3	63	0	19	34	3.234
EX5_4_2	9	0	3	3	7512.2301
EX5_4_3	17	0	9	4	4845.462
EX5_4_4	28	0	13	6	10077.7755
EX6_1_1	9	0	2	4	-0.0202
EX6_1_2	5	0	1	2	-0.03247
EX6_1_3	13	0	3	6	-0.3574
EX6_1_4	7	0	1	3	-0.2945
EX6_2_10	7	0	3	0	-3.02954
EX6_2_11	4	0	1	0	0

EX6_2_12	5	0	2	0	0.28919
EX6_2_13	7	0	3	0	-0.2162
EX6_2_14	5	0	2	0	-0.07439
EX6_2_5	10	0	3	0	-70.75
EX6_2_6	4	0	1	0	0
EX6_2_7	10	0	3	0	-0.1608
EX6_2_8	4	0	1	0	-0.027
EX6_2_9	5	0	2	0	-0.03407
EX7_2_1	8	0	2	12	1227.23
EX7_2_2	7	0	0	5	-0.38881
EX7_2_3	9	0	3	3	7048.7901
EX7_2_4	9	0	0	4	3.9511
EX7_3_1	5	0	6	1	0.3417
EX7_3_2	5	0	6	1	1.0899
EX7_3_3	6	0	6	2	0.8175
EX7_3_4	13	0	10	7	6.2746
EX7_3_4_MOD	8	0	10	2	6.2746
EX7_3_5	14	0	4	11	>1
EX7_3_5_MOD	5	0	4	2	>1
EX7_3_6	18	0	7	10	>1
EX7_3_6_MOD	10	0	7	2	>1
EX8_1_1	3	0	0	0	-2.0218
EX8_1_2	2	0	0	0	-1.0711
EX8_1_3	3	0	0	0	3
EX8_1_4	3	0	0	0	0
EX8_1_5	3	0	0	0	-1.0316
EX8_1_6	3	0	0	0	-10.086
EX8_1_7	6	0	0	5	0.0293
EX8_1_8	7	0	0	5	-0.3888
EX8_2_1	56	0	6	25	-979.186
EX8_2_4	56	0	6	75	-1197.132
EX8_3_1	116	0	17	59	0.8189924
EX8_3_10	142	0	43	65	1.546076
EX8_3_11	116	0	17	59	0.8004376
EX8_3_12	121	0	17	64	0.9993058
EX8_3_12_C	121	0	17	64	0.9993058
EX8_3_13	116	0	18	54	43.08948
EX8_3_14	111	0	17	54	2.498196
EX8_3_14_C	111	0	17	54	2.498196
EX8_3_2	111	0	27	49	0.41233
EX8_3_3	111	0	27	49	0.4166031

EX8_3_4	111	0	27	49	3.579982
EX8_3_5	111	0	27	49	0.06911967
EX8_3_6	111	0	27	49	0.5
EX8_3_7	127	0	27	65	1.232619
EX8_3_8	127	0	28	65	3.256119
EX8_3_9	79	0	18	27	0.763002
EX8_4_1	23	0	0	10	0.61857
EX8_4_2	25	0	0	10	0.485152
EX8_4_3	53	0	0	25	0.00464972
EX8_4_4	18	0	0	12	0.21246
EX8_4_5	16	0	0	11	0.0003075
EX8_4_6	15	0	0	8	0.00114
EX8_4_7	63	0	0	40	29.0473
EX8_4_8	43	0	0	30	3.32185
EX8_5_1	7	0	3	2	-0.00988
EX8_5_2	7	0	3	2	0
EX8_5_3	6	0	3	2	-0.004
EX8_5_4	6	0	3	2	-0.00033
EX8_5_5	6	0	3	2	-0.007
EX8_5_6	7	0	2	2	-0.0012
EX8_6_1R05	10	0	0	0	-9.103852
EX8_6_1R10	25	0	0	0	-28.422532
EX8_6_1R15	40	0	0	0	-52.322627
EX8_6_1R20	55	0	0	0	-77.177043
EX8_6_1R25	70	0	0	0	-102.372663
EX8_6_1R30	85	0	0	0	-128.286571
EX8_6_2R05	10	0	0	0	-9.2995
EX8_6_2R10	25	0	0	0	-31.88863
EX8_6_2R15	40	0	0	0	-63.162119
EX8_6_2R20	55	0	0	0	-97.417393
EX8_6_2R25	70	0	0	0	-136.072704
EX8_6_2R30	85	0	0	0	-177.578647
EX8_6_2R40	115	0	0	0	-268.394773
EX8_6_2R50	145	0	0	0	-366.635589
EX9_1_1	14	0	7	5	-13
EX9_1_10	15	0	7	5	-3.25
EX9_1_2	11	0	5	4	-16
EX9_1_3	30	6	27	1	29.2
EX9_1_4	11	0	5	4	-37
EX9_1_5	14	0	7	5	-1
EX9_1_6	21	6	19	1	-49

EX9_1_7	24	6	21	1	-26
EX9_1_8	15	0	7	5	-1.75
EX9_1_9	18	5	16	1	3.111
EX9_2_1	11	0	5	4	17
EX9_2_2	11	0	7	4	100
EX9_2_3	17	0	9	6	0
EX9_2_4	9	0	5	2	0.5
EX9_2_5	9	0	4	3	5
EX9_2_6	17	0	6	6	-1
EX9_2_7	11	0	5	4	17
EX9_2_8	5	0	3	2	1.5
EX9_2_9	13	3	11	1	2
EX12_2_1	6	3	3	2	7.6672
EX12_2_2	4	1	2	1	1.0765
EX12_2_3	12	4	9	4	4.5796
EX12_2_3_N	8	4	5	4	4.5796
EX12_2_4	12	8	4	3	-0.94347
EX12_2_4N	12	8	4	3	-0.94347
EX12_2_5	9	6	9	1	31
EX12_2_6	6	3	4	1	-17
EX14_1_1	4	0	0	4	0
EX14_1_2	7	0	0	9	0
EX14_1_3	4	0	0	4	0
EX14_1_4	4	0	0	4	0
EX14_1_5	7	0	4	2	0
EX14_1_6	10	0	1	14	0
EX14_1_7	11	0	0	17	0
EX14_1_8	4	0	0	4	0
EX14_1_8_N	3	0	0	0	0
EX14_1_9	3	0	0	2	0
EX14_2_1	6	0	1	6	0
EX14_2_2	5	0	1	4	0
EX14_2_3	7	0	1	8	0
EX14_2_4	6	0	1	6	0
EX14_2_5	5	0	1	4	0
EX14_2_6	6	0	1	6	0
EX14_2_7	7	0	1	8	0
EX14_2_8	5	0	1	4	0
EX14_2_9	5	0	1	4	0
Total Problems:		158			
Number of average variables:		26			

APPENDIX F Solution Statistics - Base Case

Name	Best Known Sol	Objective	Best Iteration	Best GRG Call	Total GRGCalls	Locals Found	Fcns to Best	Total Fcn Calls	Time to Best	Runtime	GAP
EX2_1_1	-17	-16.5	201	1	4	4	202	1007	0.11	0.5	2.94%
EX2_1_10	49318	49318	201	1	79	3	283	9854	1.76	10.65	0.00%
EX2_1_2	-213	-213	201	1	15	1	202	1226	0.33	1.37	0.00%
EX2_1_3	-15	-15	201	1	13	5	202	1206	0.6	1.92	0.00%
EX2_1_4	-11	-11	201	1	9	1	213	1122	0.6	1.48	0.00%
EX2_1_5	-268	-268.01	201	1	6	1	207	1147	1.21	4.56	0.00%
EX2_1_6	-39	-36	201	1	20	3	217	1240	1.04	2.41	7.69%
EX2_1_7_1	-394.8	-394.75	201	1	18	3	214	1423	11.1	16.64	0.00%
EX2_1_7_2	-884.8	-884.75	201	1	17	3	214	1398	17.3	22.68	0.00%
EX2_1_7_3	-8695	-8695	201	1	18	3	214	1425	17.2	22.69	0.00%
EX2_1_7_4	-754.8	-633.45	201	1	6	1	217	1293	8.68	12.8	16.07%
EX2_1_7_5	-4150	-4105.3	766	16	19	9	965	1260	14.8	17.3	1.09%
EX2_1_8	15639	19923	605	15	20	6	1041	1506	3.9	5.6	27.39%
EX2_1_9	-0.375	-0.375	201	1	11	3	212	1333	0.49	1.81	0.00%
EX3_1_1	7049.2	7049.2	201	1	9	2	1084	7498	0.88	4.12	0.00%
EX3_1_2	-30666	-30666	201	1	4	1	235	1184	0.16	0.55	0.00%
EX3_1_3	-310	-310	201	1	9	5	202	1062	0.33	1.1	0.00%
EX3_1_4	-4	-4	201	1	12	3	306	1652	0.33	1.1	0.00%
EX4_1_1	-7.487	-7.4873	201	1	3	1	211	1048	0.17	0.5	0.00%
EX4_1_2	-663.5	-663.5	201	1	6	1	211	1193	0.16	0.71	0.00%
EX4_1_3	-443.7	-443.67	201	1	3	1	209	1048	0.17	0.5	0.00%
EX4_1_4	0	0	201	1	9	2	202	1112	0.11	0.6	0.00%
EX4_1_5	0	0	201	1	13	2	202	1177	0.17	0.83	0.00%
EX4_1_6	7	7	201	1	9	2	210	1138	0.11	0.6	0.00%
EX4_1_7	-7.5	-7.5	201	1	5	1	208	1066	0.11	0.5	0.00%
EX4_1_8	-16.74	-16.73	201	1	5	2	316	1473	0.22	0.77	0.05%
EX4_1_9	-5.508	-5.508	201	1	13	2	227	1284	0.16	0.87	0.00%
EX5_2_4	450	450	201	1	50	1	335	7541	0.5	5.33	0.00%
EX5_2_5	3500	3500	556	21	50	43	4362	14904	6.7	16.04	0.00%
EX5_3_2	1.8642	1.8642	201	1	81	1	471	17204	1.26	11.54	0.00%
EX5_3_3	3.234	1.6878	958	48	50	0	3292	3526	18.3	18.68	inf
EX5_4_2	7512.2	7512.2	201	1	13	1	539	5492	0.61	3.52	0.00%
EX5_4_3	4845.5	4845.5	340	12	51	2	789	3188	1.71	4.95	0.00%
EX5_4_4	10078	10078	564	29	75	6	5373	14397	6.92	15.33	0.00%
EX6_1_1	-0.02	-0.0202	346	4	9	8	594	1471	0.83	1.76	-0.09%
EX6_1_2	-0.032	-0.0324	201	1	12	2	227	1436	0.38	1.48	0.07%
EX6_1_3	-0.357	-0.3525	460	11	26	25	1009	2663	2.14	4.67	1.38%

EX6_1_4	-0.295	-0.2947	203	3	7	4	543	2092	0.82	2.47	-0.06%
EX6_2_10	-3.03	-3.052	201	1	14	2	245	2215	0.77	4.17	-0.74%
EX6_2_11	0	2E-06	201	1	10	2	229	1403	0.55	2.09	0.00%
EX6_2_12	0.2892	0.2892	201	1	10	2	232	1466	0.44	1.65	0.00%
EX6_2_13	-0.216	-0.2162	201	1	24	22	202	1565	0.72	3.85	0.00%
EX6_2_14	-0.074	-0.6954	201	1	9	2	207	1103	0.44	1.54	-834.75%
EX6_2_5	-70.75	-70.752	549	29	57	48	2707	4834	4.34	7.75	0.00%
EX6_2_6	0	-3E-06	201	1	11	2	211	1421	0.55	2.19	0.00%
EX6_2_7	-0.161	-0.1609	201	1	12	3	272	2175	0.88	4.34	-0.03%
EX6_2_8	-0.027	-0.027	201	1	12	3	235	1472	0.55	2.2	-0.02%
EX6_2_9	-0.034	-0.0341	201	1	14	4	276	1902	0.6	2.19	0.01%
EX7_2_1	1227.2	1226.8	201	1	6	2	429	8870	0.55	3.68	-0.04%
EX7_2_2	-0.389	-0.3888	201	1	12	7	242	1901	0.16	1.09	0.00%
EX7_2_3	7048.8	7048.9	276	3	4	4	1074	2256	0.99	2.03	0.00%
EX7_2_4	3.9511	3.9179	202	2	13	10	894	6643	0.55	3.79	-0.84%
EX7_3_1	0.3417	0.3417	644	13	13	11	712	1068	0.83	0.99	0.01%
EX7_3_2	1.0899	1.0899	201	1	3	1	206	1048	0.55	1.43	0.00%
EX7_3_3	0.8175	0.8175	201	1	9	2	211	1503	0.38	1.37	0.00%
EX7_3_4	6.2746	6.2746	201	1	72	1	367	23995	1.1	14.39	0.00%
EX7_3_4_MOI	6.2746	6.2747	201	1	3	1	209	1086	0.61	1.27	0.00%
EX7_3_5	>1	5.6897	201	1	23	0	672	11855	0.99	8.24	inf
EX7_3_5_MOI	>1	2.7416	376	10	39	1	1017	3422	0.93	2.53	ok
EX7_3_6	>1	0	201	1	30	0	226	1830	1.43	6.59	inf
EX7_3_6_MOI	>1	0	621	5	5	0	661	1040	3.02	4.45	inf
EX8_1_1	-2.022	-2.0218	201	1	5	1	210	1074	0.16	0.49	0.00%
EX8_1_2	-1.071	-1.0709	201	1	8	3	208	1084	0.16	0.6	0.02%
EX8_1_3	3	3	201	1	2	1	226	1054	0.22	0.49	0.00%
EX8_1_4	0	0	201	1	9	1	202	1125	0.11	0.6	0.00%
EX8_1_5	-1.032	-1.0316	201	1	3	2	218	1058	0.11	0.44	0.00%
EX8_1_6	-10.09	-10.086	363	3	6	2	409	1111	0.27	0.6	0.00%
EX8_1_7	0.0293	0.0293	297	2	11	2	658	3623	0.38	1.87	0.03%
EX8_1_8	-0.389	-0.3888	201	1	12	7	242	1901	0.22	1.1	0.00%
EX8_2_1	-979.2	-979.18	222	2	41	5	447	4975	4.78	15.22	0.00%
EX8_2_4	-1197	-1197.1	201	1	12	5	377	3013	5.82	17.69	0.00%
EX8_3_1	0.819	0.774	967	26	26	24	25071	25104	125	125.9	5.49%
EX8_3_10	1.5461	1.4546	428	13	25	18	2249	6834	77.6	169	5.92%
EX8_3_11	0.8004	0.7317	913	17	19	11	22529	34766	126	141.4	8.58%
EX8_3_12	0.9993	-0.3017	720	3	3	0	2928	3208	71	96.73	inf
EX8_3_12_car	0.9993	0.9944	201	1	47	33	1588	61396	39.3	173.3	0.49%
EX8_3_13	43.089	42.712	219	3	22	9	11630	54972	39.8	164	0.88%
EX8_3_14	2.4982	2.2031	414	6	68	5	7017	46634	51.9	178.2	11.81%
EX8_3_14_car	2.4982	-0.0001	201	1	1	0	340	1139	27	72.83	inf

EX8_3_2	0.4123	0.4053	201	1	17	17	1377	9220	25.6	114.4	1.70%
EX8_3_3	0.4166	0.4166	645	8	11	11	7597	12082	71.1	106.2	0.00%
EX8_3_4	3.58	3.5725	623	24	30	29	28019	40459	90.9	134.1	0.21%
EX8_3_5	0.0691	0.0667	379	4	33	33	1465	19950	51.1	137.2	3.51%
EX8_3_6	0.5	0.499	873	30	33	26	15454	18102	113	119.4	0.19%
EX8_3_7	1.2326	0.9856	201	1	20	11	384	9410	34.4	118.1	20.04%
EX8_3_8	3.2561	3.2311	695	18	41	40	35763	80261	156	242.2	0.77%
EX8_3_9	0.763	0.7615	399	19	79	78	5469	25250	22.6	58.49	0.20%
EX8_4_1	0.6186	0.6186	201	1	9	1	257	1726	0.27	1.2	0.00%
EX8_4_2	0.4852	0.4851	201	1	4	1	300	1402	0.33	1.1	-0.01%
EX8_4_3	0.0046	0.0046	201	1	10	4	260	3499	1.05	5.33	0.00%
EX8_4_4	0.2125	0.2125	201	1	11	1	293	2168	0.28	1.43	0.00%
EX8_4_5	0.0003	0.0003	201	1	23	19	338	3103	0.33	2.47	0.01%
EX8_4_6	0.0011	0.0012	284	4	56	47	3099	36514	1.81	21.25	3.93%
EX8_4_7	29.047	29.054	201	1	17	1	345	12192	2.15	14.67	0.02%
EX8_4_8	3.3219	3.322	389	2	16	1	8682	68533	6.43	47.19	0.00%
EX8_5_1	-0.01	-0.0005	327	7	15	14	913	2505	1.32	3.51	95.41%
EX8_5_2	0	-0.0014	284	2	3	3	551	1573	0.87	2.74	-0.14%
EX8_5_3	-0.004	-0.0041	201	1	11	3	223	1551	0.55	2.26	-3.16%
EX8_5_4	-3E-04	-0.0005	209	2	17	7	324	1873	0.72	2.97	-64.27%
EX8_5_5	-0.007	-0.011	890	6	8	3	1374	1694	1.92	2.31	-57.33%
EX8_5_6	-0.001	-0.0012	201	1	9	2	283	1706	0.77	2.86	2.83%
EX8_6_1R05	-9.104	-9.1039	201	1	5	3	312	2676	0.33	2.47	0.00%
EX8_6_1R10	-28.42	-28.423	349	5	14	14	2563	7615	5.99	17.58	0.00%
EX8_6_1R15	-52.32	-52.323	407	11	56	56	7431	34265	33.5	152.1	0.00%
EX8_6_1R20	-77.18	-75.589	302	6	38	38	6441	29584	53	238.4	2.06%
EX8_6_1R25	-102.4	-99.369	999	45	45	45	41678	41679	530	530.7	2.93%
EX8_6_1R30	-128.3	-125.85	682	36	66	66	47232	83730	854	1510	1.90%
EX8_6_2R05	-9.3	-9.2995	201	1	9	6	342	2174	0.33	1.76	0.00%
EX8_6_2R10	-31.89	-31.889	284	2	3	3	922	2067	1.65	3.62	0.00%
EX8_6_2R15	-63.16	-63.162	201	1	13	13	533	6035	2.14	19.28	0.00%
EX8_6_2R20	-97.42	-97.417	201	1	38	38	673	17747	4.72	89.04	0.00%
EX8_6_2R25	-136.1	-136.07	628	11	12	12	6400	7314	52.3	61.02	0.00%
EX8_6_2R30	-177.6	-177.58	886	34	46	46	22149	29234	252	330.7	0.00%
EX8_6_2R40	-268.4	-267.62	560	13	15	15	11195	13024	234	284.3	0.29%
EX8_6_2R50	-366.6	-366.62	750	24	41	41	23140	38490	805	1313	0.00%
EX9_1_1	-13	-13	201	1	4	1	211	1044	1.05	3.24	0.00%
EX9_1_10	-3.25	-3.25	201	1	34	12	209	1243	0.66	3.18	0.00%
EX9_1_2	-16	-16	201	1	5	2	202	1016	0.66	2.25	0.00%
EX9_1_3	29.2	-29.2	201	1	66	61	202	1230	34.2	142.7	0.00%
EX9_1_4	-37	-37	201	1	3	1	202	1016	0.88	2.58	0.00%
EX9_1_5	-1	-1	201	1	20	19	202	1041	0.66	2.64	0.00%

EX9_1_6	-49	-49	201	1	37	33	202	1083	18.8	77.66	0.00%
EX9_1_7	-26	-26	227	4	46	34	261	1341	38.3	74.69	0.00%
EX9_1_8	-1.75	-3.25	201	1	34	12	209	1243	0.6	3.13	-85.71%
EX9_1_9	3.111	3.1111	201	1	15	10	202	1071	8.96	37.74	0.00%
EX9_2_1	17	17	290	2	26	23	293	1061	1.1	3.51	0.00%
EX9_2_2	100	99.979	201	1	31	3	207	1216	0.55	2.75	-0.02%
EX9_2_3	0	0	201	1	3	2	202	1009	1.26	4.06	0.00%
EX9_2_4	0.5	0.5	201	1	34	1	210	1357	0.49	2.69	0.00%
EX9_2_5	5	5	201	1	3	2	211	1031	0.71	2.41	0.00%
EX9_2_6	-1	-1	201	1	9	1	202	1055	0.71	2.52	0.00%
EX9_2_7	17	17	290	2	26	23	293	1061	1.1	3.52	0.00%
EX9_2_8	1.5	1.5	201	1	5	1	202	1020	0.28	1.05	0.00%
EX9_2_9	2	2	201	1	8	3	202	1035	1.92	7.57	0.00%
EX12_2_1	7.6672	7.6671	249	4	15	7	271	1112	1.6	4.23	0.00%
EX12_2_2	1.0765	1.0764	201	1	8	1	208	1057	0.33	0.94	-0.01%
EX12_2_3	4.5796	5.2728	786	13	20	8	1027	1454	5.88	6.87	15.14%
EX12_2_3_N	4.5796	4.5796	848	22	26	11	1313	1632	4.23	4.89	0.00%
EX12_2_4	-0.943	-0.9143	202	2	26	7	265	1845	12	16.54	3.09%
EX12_2_4N	-0.943	-0.9435	240	9	24	8	266	1071	14	14.17	0.00%
EX12_2_5	31	31	201	1	10	4	202	1078	5.49	22.52	0.00%
EX12_2_6	-17	-17	201	1	15	5	202	1205	1.04	3.62	0.00%
EX14_1_1	0	-2E-07	201	1	7	4	221	1198	0.11	0.55	0.00%
EX14_1_2	0	8E-21	201	1	36	2	480	9980	0.33	5.38	0.00%
EX14_1_3	0	-1E-19	201	1	10	4	231	1556	0.17	0.88	0.00%
EX14_1_4	0	-2E-20	201	1	14	5	235	1951	0.16	1.15	0.00%
EX14_1_5	0	0	201	1	7	2	202	1092	0.77	2.42	0.00%
EX14_1_6	0	7E-21	201	1	8	3	290	2174	0.88	3.35	0.00%
EX14_1_7	0	0.135	209	2	22	13	1105	10588	0.77	6.92	13.50%
EX14_1_7_N	0	0.0221	755	20	25	6	15380	18978	10.1	12.47	2.21%
EX14_1_8	0	0.0414	201	1	2	1	272	1167	0.16	0.49	4.14%
EX14_1_8_N	0	0.0001	363	6	9	2	536	1229	0.38	0.71	0.01%
EX14_1_9	0	0	201	1	8	2	215	1127	0.11	0.6	0.00%
EX14_2_1	0	0	201	1	2	1	247	1098	0.6	1.76	0.00%
EX14_2_2	0	0	201	1	7	1	222	1154	0.5	1.48	0.00%
EX14_2_3	0	6E-07	201	1	4	1	240	1220	0.77	2.52	0.00%
EX14_2_4	0	7E-20	201	1	3	1	271	1179	0.71	1.97	0.00%
EX14_2_5	0	0	201	1	3	1	225	1072	0.49	1.48	0.00%
EX14_2_6	0	0	201	1	3	1	245	1193	0.71	1.97	0.00%
EX14_2_7	0	4E-20	201	1	6	1	233	1265	0.82	3.13	0.00%
EX14_2_8	0	0	201	1	8	1	230	1180	0.55	1.54	0.00%
EX14_2_9	0	0	201	1	4	1	219	1063	0.55	1.65	0.00%

Bibliography

Biegler, L.T., I.E. Grossman, and A. W. Westerberg, 1997, *Systematic Methods of Chemical Process Design*, Prentice-Hall, Englewood Cliffs, NJ.

Boender, C.G.E., H.E. Romeijn, 1995, "Stochastic Methods", in Handbook of Global Optimization, pp. 829-869, R. Horst, P.M. Pardalos, eds., Kluwer Academic Publishers.

Brooke, A. et al., 1992, *GAMS: A User's Guide*, Boyd and Fraser, Danvers, MA.

Dixon, L., G.P. Szego, 1975, "Towards Global Optimization", in *Proceedings of a Workshop at the University of Cagliari, Italy*, North Holland.

Drud, A., 1994, "CONOPT—A Large-Scale GRG-Code", *ORSA Journal on Computing*, Vol. 6, No. 2.

Edgar, T.F., D.M. Himmelblau, L.S. Lasdon, 2001, *Optimization of Chemical Processes*, the McGraw-Hill Companies, Inc.

Floudas, C., 1995, *Nonlinear and Mixed-Integer Optimization*, Oxford University Press, New York.

Floudas, C.A., et. al., 1999, *Handbook of Test Problems in Local and Global Optimization*, Kluwer Academic Publishers.

Glover, F., 1998, "A Template for Scatter Search and Path Relinking", in *Artificial Evolution, Lecture Notes in Computer Science* 1363, J.-K Hao, E. Lutton, E. Ronald, M. Schoenauer and D. Snyers, eds., Springer Verlag, pp. 13-54.

Horst, R., P.M. Pardalos, 1995, *Handbook of Global Optimization*, Kluwer Academic Publishers.

Horst, R., H. Tuy, 1993, *Global Optimization: Deterministic Approaches*, Springer-Verlag, Berlin, 2nd rev. edition.

Laguna, M., Rafael Marti, 2000, "Experimental Testing of Advanced Scatter Search Designs for Global Optimization of Multimodal Functions", working paper, Department D'Estadistica i Investigacio Operativa, Universitat de Valencia, Burjassot 46100, Spain.

Laguna, M., Rafael Marti, 2001, "The OptQuest Callable Library", to appear in *Optimization Software Class Libraries,* Stefan Voss and D. Woodruff, eds., Kluwer Academic Publishers, Boston.

Locatelli, M., F. Schoen, 1999, "Random Linkage: a Family of Acceptance/Rejection Algorithms for Global Optimization", *Math. Programming*, 85, 2, pp. 379-396.

Murtagh, B.A., M.A. Saunders, 1982, "A Projected Lagrangian Algorithm and Its Implementation for Sparse Nonlinear Constraints", *Mathematical Programming Study*, 16, pp. 84-117.

Nash, S.G., A. Sofer, 1996, *Linear and Nonlinear Programming*, the McGraw-Hill Companies, Inc.

Nash, S.G., 1998, "Nonlinear Programming", *OR/MS Today*, pp. 36-45.

Nocedal, J., S. J. Wright, 1999, *Numerical Optimization*, Springer Series in Operations Research.

Pardalos, P.M., H.E. Romeijn, H. Tuy, 2000, "Recent Developments and Trends in Global Optimization", *Journal of Computational and Applied Mathematics*, v.124, 1-2, pp. 209-228.

Pintér, J.D., 1996, *Global Optimization in Action*, Kluwer Academic Publishers.

Rinnooy Kan, A.H.G., G.T. Timmer, 1987, "Stochastic Global Optimization Methods; part I: Clustering Methods", *Mathematical Programming*, 37, pp. 27-56.

Rinnooy Kan, A.H.G., G.T. Timmer, 1987, "Stochastic Global Optimization Methods; part II: Multi Level Methods", *Mathematical Programming*, 37, pp. 57-78.

Smith, S., L. Lasdon, 1992, "Solving Large Sparse Nonlinear Programs Using GRG", *ORSA Journal on Computing*, Vol. 4, No. 1, pp. 3-15.

Ugray, Cs., 1979, "Valuation of Some Random Search Methods", *Math. Operationsforsch. Statist., Ser. Optimization*, Vol. 10, No. 1, pp.57-65.

VITA

Zsolt Gyula Ugray was born in Budapest, Hungary on July 29, 1967, the son of Csilla and László Ugray. After completing his studies at Váci Utcai Általános Iskola in 1981 and at Veres Pálné Gimnázium in 1985 (his elementary and high schools, respectively), he served one year in the military. He entered the Technical University of Budapest in the Fall of 1986. During the Fall of 1990, he attended the College of Charleston, South Carolina. He received the degree of Diploma in Electrical Engineering (equivalent of Master of Science in Electrical Engineering, MSEE) from the Technical University of Budapest in 1991. That Fall he entered the Graduate School of the University of Texas at Arlington. In the Fall of 1993 he entered the Graduate School of Business at the University of Texas at Austin. From the Summer of 1997 until the end of 1998 he worked as an engineer at Fourth State Technology. From January 1999 until the Fall of 2001 he was employed by Advanced Energy Industries as a Senior Software Engineer.

Permanent Address: 1207 Ridgehaven Dr., Austin, TX 78723

This dissertation was typed by the author.