Chapter 2

A MULTISTART SCATTER SEARCH HEURISTIC FOR SMOOTH NLP AND MINLP PROBLEMS

Zsolt Ugray[1], Leon Lasdon[2], John C. Plummer[3], Fred Glover[4], Jim Kelly[5] and Rafael Martí[6]

[1]*Business Information Systems Department, Utah State University, 3515 Old Main Hill Logan, UT 84322-3515, zsolt.ugray@ucr.edu*

[2]*The University of Texas at Austin, McCombs School of Business, Management and Information Systems Dept. B6500, Austin, TX 78712-0212, asdon@mail.utexas.edu*

[3]*Dept of CIS/QMST, Texas State University, 601 University Dr, San Marcos, TX 78666, jcplummer@mail.utexas.edu*

[4]*Leads School of Business, University of Colorado, Boulder, CO 80309-0419, fred.glover@colorado.edu*

[5]*OptTek Systems, Inc 1919 Seventh St., Boulder, CO 80302, Kelly@opttek.com*

[6]*Departamento de Estadística e I.O., Facultad de Matemáticas, Universitat de Valencia, Dr. Moliner 50, 46100 Burjassot, Valencia, Spain, rafael.marti@uv.es*

Abstract: The algorithm described here, called OptQuest/NLP or OQNLP, is a heuristic designed to find global optima for pure and mixed integer nonlinear problems with many constraints and variables, where all problem functions are differentiable with respect to the continuous variables. It uses OptQuest, a commercial implementation of scatter search developed by OptTek Systems, Inc., to provide starting points for a gradient-based local NLP solver. This solver seeks a local solution from a subset of these points, holding discrete variables fixed. The procedure is motivated by our desire to combine the superior accuracy and feasibility-seeking behavior of gradient-based local NLP solvers with the global optimization abilities of OptQuest. Computational results include 144 smooth NLP and MINLP problems due to Floudas et al, most with both linear and nonlinear constraints, coded in the GAMS modeling language. Some are quite large for global optimization, with over 100 variables and many constraints. Global solutions to almost all problems are found in a small number of NLP solver calls, often one or two.

Keywords: Global Optimization, Multistart Heuristic, Mixed Integer Nonlinear Programming, Scatter Search, Gradient Methods

1. Introduction

This paper describes a multistart heuristic algorithm designed to find global optima of smooth constrained nonlinear programs (NLPs) and mixed integer nonlinear programs (MINLPs). It uses the widely used scatter search software OptQuest (Laguna and Martí, 2000) to generate trial points, which are candidate starting points for a local NLP solver. These are filtered to provide a smaller subset from which the local solver attempts to find a local optimum. Our implementation uses the generalized reduced gradient NLP solver GRG (Smith and Lasdon, 1993), but in principle any local NLP solver can be used, including ones that do not require derivatives. However, we focus on gradient-based solvers because they are by far the most widely used, and currently are the only ones capable of solving NLPs with hundreds or thousands of variables and constraints.

- Problem Statement
  The most general problem this algorithm can solve has the form

  minimize $f(x,y)$ (1.1)

  subject to the nonlinear constraints

  $$gl \le G(x,y) \le gu \quad (1.2)$$

  the linear constraints

  $$l \le A_1 x + A_2 y \le u \quad (1.3)$$

  $$x \in S, y \in Y \quad (1.4)$$

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, 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 we assume that it is closed and bounded, i.e., that 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 needed in order that a gradient-based local NLP solver be applicable to relaxed NLP subproblems formed from (1)-(3) by allowing the y variables to be continuous.

- Multi-start Algorithms

In this section, which reviews past work on multi-start algorithms, we focus on unconstrained problems where there are no discrete variables, since to the best of our knowledge multi-start algorithms have been investigated theoretically only in this context. These problems have the form

$$\text{minimize } f(x) \tag{1.5}$$

$$\text{subject to } x \in S \tag{1.6}$$

where all global minima of f are assumed to occur in the interior of S. By multi-start we mean any algorithm that attempts to find a global solution to (1.5)-(1.6) by starting a local NLP solver, denoted by L, from multiple starting points in S. The most basic multi-start method generates uniformly distributed points in S, and starts L from each of these. This is well known to converge to a global solution with probability one as the number of points approaches infinity--in fact, the best of the starting points converges as well. This procedure is very inefficient because the same local solution is located many times. A convergent procedure that largely overcomes this difficulty is called multi-level single linkage (MLSL) (Rinnooy Kan and Timmer, 1987). This uses a simple rule to exclude some potential starting points. A uniformly distributed sample of N points in S is generated, and the objective, f, is evaluated at each point. The points are sorted according to their f values, and the qN best points are retained, where q is an algorithm parameter between 0 and 1. L is started from each point of this reduced sample, except if there is another sample point within a certain critical distance which has a lower f value. L is also not started from sample points which are too near the boundary of S, or too close to a previously discovered local minimum. Then, N additional uniformly distributed points are generated, and the procedure is applied to the union of these points and those retained from previous iterations. The critical distance referred to above decreases each time a new set of sample points is added. The authors show that, if the sampling continues indefinitely, each local minimum of f will be located, but the total number of local searches is finite with probability one. They also develop Bayesian stopping rules, which incorporate assumptions about the costs and

potential benefits of further function evaluations, to determine when to stop the procedure.

When the critical distance decreases, 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. This 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. Recently, Locatelli and Schoen (1999) introduce 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 nondecreasing function $\phi(d)$, where d is the distance from the current sample point to the closest of the previous sample points with a better function value. Assumptions on this function that give RL methods the same theoretical properties as MLSL are derived in the above reference.

Recently, Fylstra et al. have implemented a version of MLSL which can solve constrained problems (Frontline Systems, Inc., 2000). See also www.frontsys.com. Limited to problems with no discrete variables *y,* it uses the L_1 exact penalty function, defined as

$$P_1(x,w) = f(x) + \sum_{i=1}^{m} w_i viol(g_i(x)) \tag{1.7}$$

where the w_i are nonnegative penalty weights, $m = m_1 + m_2$, and the vector *g* has been extended to include the linear constraints (1.4). 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 (1.1)-(1.4), u^* is a corresponding optimal multiplier vector, and the second order sufficiency conditions are satisfied at (x^*, u^*), then if

$$w_i > abs(u_i^*) \tag{1.8}$$

x^* is a local unconstrained minimum of P_1. If (1.1)-(1.4) has several local minima, 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 minimum at each of these local constrained minima. 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. We will use P_1 in the same way in our OQNLP algorithm. We are not aware of any theoretical investigations of this extended MLSL

procedure, so it must currently be regarded as a heuristic. Comparative testing of this extension of MLSL and OQNLP is planned.

2. Scatter Search and the Optquest Callable Library

We provide only a brief description of Scatter Search and its implementation in the OptQuest callable library here, since it is discussed in Laguna and Martí (2001) and in Laguna and Martí (2000). See also www.opttek.com. Like genetic algorithms (GAs) and other meta-heuristics, scatter search operates on a set of solutions, called the *population* in the GA literature and the *reference set* in scatter search papers, that is maintained and updated from iteration to iteration. Scatter search (SS) is a novel instance of evolutionary methods, because it violates the premise that evolutionary approaches must be based solely on randomization. SS is also novel, in comparison to GAs, by being founded on strategies that were proposed as augmentations to GAs more than a decade after their debut in scatter search. Scatter search embodies principles and strategies that are still not emulated by other evolutionary methods, and that prove advantageous for solving a variety of complex optimization problems.

Scatter Search is designed to operate on a set of points, called reference points, which constitute good solutions obtained from previous solution efforts. Notably, the basis for defining "good" includes special criteria such as diversity that purposefully go beyond the objective function value. The approach systematically generates combinations of the reference points to create new points, each of which may (optionally) be mapped into an associated feasible point. The underlying combination mechanism uses linear combinations.

OptQuest is available as a 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) create an initial reference set, (d) retrieve a trial solution to be evaluated and, (e) communicate these objective and constraint values back to OptQuest, which uses them as the input to update the reference set. For a complete description, see Laguna and Martí (2001).

3. Gradient-Based NLP Solver and GRG

There are many papers and texts discussing gradient-based NLP solvers, e.g., Nash and Sofer (1996), Nocedal and Wright (1999), Edgar, Himmelblau and Lasdon (2001). These solve problems of the form (1.1)-(1.4), but with no discrete (y) variables. They require a starting point as input, and use values and gradients of the problem functions to generate a sequence of points which, under fairly general smoothness and regularity conditions, converges to a local optimum. 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. 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 1.e-4) determines when a constraint is satisfied. If the constraint has the form $g(x) \geq l$, it is considered satisfied in the GRG code used here if

$$abs(g(x) - l) \geq -ft(1.0 + abs(l)).$$

The optimality tolerance, *ot,* (default value 1.e-4) and a number of consecutive iterations, *nstop,* (default value 10) determine when the current point is declared optimal. This occurs when

$$kterrnorm \leq ot$$

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

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

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

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 our computational testing, and we discuss this in more detail later.

Let L be a local NLP solver capable of solving (1.1)-(1.4), 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:

$$B(x_i^*) = \{x_0 \mid x_0 \in S, L(x_0) = x_i^*\}. \tag{3.1}$$

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, as it is for convex programs, and problem difficulty increases as this relative volume approaches zero.

4. Comparing Search Methods and Gradient-Based NLP Solvers

For smooth problems, the relative advantages of a search method like OptQuest over a gradient-based NLP solver are its ability to locate an approximation to a good local solution (often the global optimum), and the fact that it can handle discrete variables. Gradient-based NLP solvers converge to the "nearest" local solution, and have no facilities for discrete variables, unless they are imbedded in a rounding heuristic or branch-and-bound method. Relative disadvantages of search methods are their limited accuracy, and their weak abilities to deal with equality constraints (more generally, narrow feasible regions). They find it difficult to satisfy many nonlinear constraints to high accuracy, but this is a strength of gradient-based NLP solvers. Search methods also require an excessive number of iterations to find approximations to local or global optima accurate to more than 2 or 3 significant figures, while gradient-based solvers usually achieve 4 to 8-digit accuracy rapidly.

The motivation for combining search and gradient-based solvers in a multi-start procedure is to achieve the advantages of both while avoiding the disadvantages of either. Surprisingly, we have been unable to locate any published efforts in this direction, besides the Frontline extended MLSL method discussed in Section 2.

5. The OQNLP Algorithm

A pseudo-code description of the simplest OQNLP algorithm follows:

- INITIALIZATION
  Read_Problem_Parameters (*n, p*, m_1, m_2 , bounds, starting point);
  Setup_OptQuest_Parameters (size, iteration limits, population,
  accuracy, variables, bounds, constraints);
  Initialize_OptQuest_Population;

- STAGE 1: INITIAL OPTQUEST ITERATIONS AND FIRST GRG CALL
  WHILE (unevaluated trial points from initial population remain) **DO**
  {
  Get (trial solution from OptQuest);
  Evaluate (objective and constraint values at trial solution,);
  Put (trial solution , objective and constraint values to OptQuest
  database);
  }
  ENDDO
  Get_Best_Point_from_OptQuest_database (starting point);
  Call_GRG (starting point, local solution);
  threshold = P_1 value of local solution;

- STAGE 2: MAIN ITERATIVE LOOP
  WHILE (stopping criteria not met) **DO**
  {
  Get (trial solution from OptQuest);
  Evaluate (objective and constraint values at trial solution,);
  Put (trial solution, objective and constraint values to OptQuest database);
  Calculate_ Penalty_ Function (trial solution, P_1);
  IF (distance and merit filter criteria are satisfied) **THEN**
  {
  Call_GRG (trial solution, local solution);
  Analyze_Solution (GRG Terminating Condition);
  Update_Local_Solutions_Found;
  Update_Largest_Lagrange_Multipliers_Found;
  }
  ELSE IF (P_1 > threshold for *waitcycle* consecutive iterations) increase *threshold*
  }
  ENDDO

After initialization, there are two main stages. In the "initial OptQuest iterations" stage, the objective and constraint values at all trial points generated by the initial OptQuest population (including the

population points themselves) are evaluated, and these values are returned to OptQuest, which computes its penalty function, *P,* at each point. The point with the best *P* value is selected, and GRG is started from this point. If there are any discrete variables, *y*, they are fixed at their current values during the GRG solution. In general, the trial points are scattered within the rectangle defined by the bounds on the variables, so choosing the best corresponds to performing a coarse search over this rectangle. If the best point falls inside the basin of attraction of the global optimum relative to the GRG solver (as it often does), then if the subsequent GRG call is successful, it will find a global optimum. This call also determines optimal Lagrange multiplier values, u^*, for the constraints. These are used to determine initial values for the penalty weights, w_i, satisfying (1.8), which are used in the exact penalty function, P_1, defined in (1.7). All local optima found are stored in a linked list, along with the associated Lagrange multipliers and objective values. Whenever a new local optimum is found, the penalty weights are updated so that (1.8) is satisfied over all known local optima.

The main iterative loop of stage 2 obtains trial points from OptQuest, and starts GRG from the subset of these points determined by two filters. The distance filter helps insure that the GRG starting points are diverse, in the sense that they are not too close to any previously found local solution. Its goal is to prevent GRG from starting more than once within the basin of attraction of any local optimum, so it plays the same role as the rule in the MLSL algorithm of Section 2, which does not start at a point if it is within a critical distance of a better point. 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,* GRG 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 GRG is never started.

The merit filter helps insure that the GRG starting points have high quality, by not starting from candidate points whose exact penalty function value P_1 (see (1.7)) is greater than a threshold. This threshold is set initially to the P_1 value of the best candidate point found in the first stage of the algorithm. If trial points are rejected by this test for more than *waitcycle* consecutive iterations, the threshold is increased by the updating rule:

$$\text{threshold} \leftarrow \text{threshold} + \text{thfact}*(1.0+\text{abs}(\text{threshold})) \qquad (4.2)$$

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. When a trial point is accepted by the merit filter, *threshold* is decreased by setting it to the P_1 value of that point.

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 2.1, which shows the stationary point at the origin and the 6 local minima of a 2 variable unconstrained function (called the six-hump camelback function) as dark squares, labeled with their objective value. The ten points from which OQNLP 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 in stage 1, and 10 points in the initial population. 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 two times. The other global optimum at (-.0898, .7127) is found first at iteration 268, and is located 6 times.

The limit on total OQNLP iterations in this run was 1000. GRG was started at only 9 of the 846 OptQuest trial points generated in the main iterative loop of stage 2. 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 rejected 498 points, the distance filter alone 57, and both rejected 281.

Figure 2.2 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.

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 2.3 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 merit 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 one of the 2 global optima, and they are rejected by the distance filter.

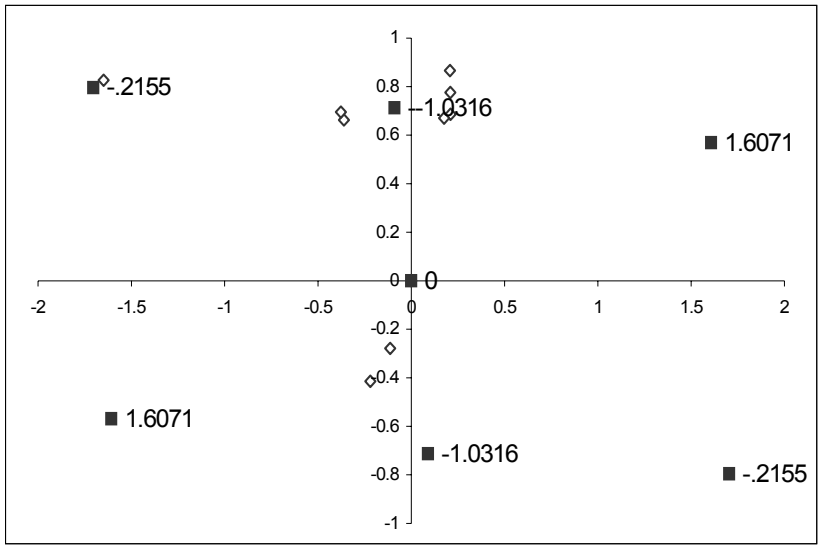

Figure 2.1. Local Optima and 10 GRG starting points for Six-Hump Camelback function

This simple example illustrates a number of important points:

1. Setting the bounds on the continuous or discrete variables to be too large in magnitude is likely to slow the OQNLP algorithm (or any search algorithm) and may lead to a poorer final solution. In the above example, if the variable bounds had been [-2,2] rather than [10,10], the trial points generated by the initial population would have had much lower objective values. OptQuest can overcome this when the initial population is updated.
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.
3. The best trial point generated by the initial reference set 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.

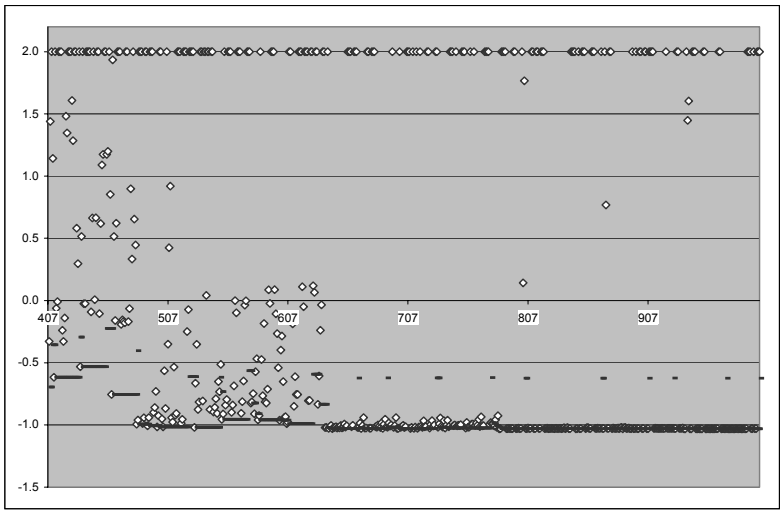

Figure 2.2. Objective and threshold values for Six-Hump Camelback function for iterations 155 to 407

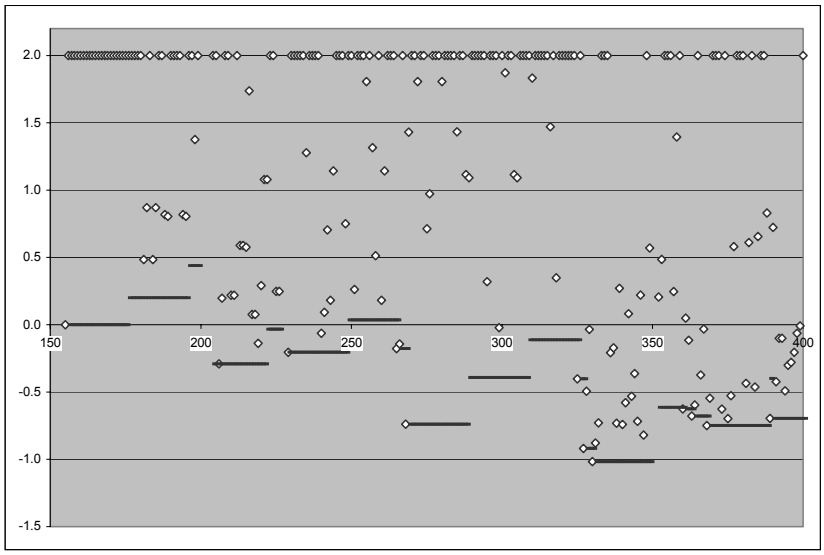

Figure 2.3. Objective and threshold values for Six-Hump Camelback function: iterations 408 to 100

Filtering Logic for Problems with Discrete Variables

The filtering logic described above must be extended when there are discrete variables (the y variables in the problem statement (1.1)-(1.4)). When a trial point (xt, yt) provided by OptQuest passes the two filtering tests and is passed to GRG, xt acts as a starting point and is changed by GRG, but the yt values are fixed and are not changed. Each new set of yt values defines a different NLP for GRG to solve, say NLP(yt), with its own set of local minima in x space, so both filters must be made specific to NLP(yt). For the distance filter, it is irrelevant if xt is close to any local minima (in x space) previously found which correspond to problems NLP(y) with y different from yt. Hence the distance filter is based on the distance from xt to local minima of NLP(yt) only. Similarly, the tests and threshold values in the merit filter must be specific to the problem NLP(yt) currently being solved. However, the weights w in the exact penalty function $P_1(x,y,w)$ used in the merit filter are based on the maximum absolute multipliers over *all* local optima for *all* vectors yt, because these weights are large enough to ensure that this function is exact for all problems NLP(y).

Therefore, in stage 2 of the algorithm, the exact penalty function, $P_1(xt,yt,w)$, is calculated at each trial point (xt,yt), and GRG is started at (xt,yt) if P_1 is smaller than the current threshold for NLP(yt). This threshold is initialized to plus infinity, so if the values yt have not occurred in a previous stage 2 trial point, GRG will be called at this point. This leads to many more GRG calls in problems with discrete variables, as we show later in the computational results sections.

This OQNLP algorithm should be regarded as a base case from which extensions will be explored and compared. The most significant of these involves the return of information from GRG to OptQuest, which is absent in the above procedure, i.e. local solutions found by GRG are not returned to OptQuest. Such solutions are generally of very high quality, and might aid the search process if they were incorporated into the OptQuest population, because at least a subset would likely be retained there. However, this should be done so as to preserve the diversity of the population. We discuss this option further in Section 9.

6. C Language Implementation, Games Interface, and the Floudas Test Problem Set

The algorithm described in the previous section has been implemented as a callable C-language function. In this form, the user supplies a C function that evaluates the objective and constraint functions, an optional routine that evaluates their first partial derivatives (finite difference approximations are used otherwise), and a calling program that supplies problem size, bounds, and an initial point, and invokes the algorithm. Algorithm parameters and options are in an options text file. We have developed an interface between this 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 enables us to invoke the OptQuest option which maps each trial point (generated as described in Section 2) into a point which satisfies the linear constraints. This is done by solving a linear program that minimizes the L_1 distance between the trial point and the feasible region defined by 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 this GAMS interface 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/ or from www.gams.com, linking to gams world and then global. Characteristics of 142 of these problems (excluding the 8_6_1 and 8_6_2 sets) are contained in Table 2.1.

Most of these problems arise from chemical engineering, but some are from general problem classes. Most are small, but a few have over 100 variables and comparable numbers of constraints, and some 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, but many problems have nonlinear constraints, and these are often the source of the nonconvexities. For example, there are many problems arising from pooling and blending applications with bilinear constraints. The best known objective value and (in most cases) the corresponding variable values are provided in Floudas, et al. (1999). 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, so there are 3N variables.

Table 2.1. Characteristics of Floudas GAMS test problems

SERIES	PROBLEMS	MAX VARS	MAX DISCRETE VARS	MAX LINEAR CONSTRAINS	MAX NONLINEAR CONSTRAINS	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 from 4 to 147	3N	0	0	0	Lenard-Jones energy min
EX8_6_2	N from 5 to 80	3N	0	0	0	Morse energy min
EX9_1_x	10	29	0	27	5	bilevel LP
EX9_2_x	9	16	0	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					

7. Computational Results on the Floudas Set of Test Problems

This section describes the results obtained when the OQNLP algorithm described in Section 6 is applied to the Floudas GAMS test problems. The main algorithm parameters and options used are shown in Table 2.2 below.

Table 2.2. OptQuest, GRG, and OQNLP parameters and options used

OptQuest and OQNLP Parameters	GRG Parameters
Use linear constraints = yes	Feasibility tolerance = 1.e-4, except series 8_3_x uses 1.e-6
Total iterations = 1000	Optimality tolerance = 1.e-4, except series 8_3_x uses 1.e-6
Total stage 1 iterations = 200	Consecutive iterations for fractional change termination = 20
Waitcycle = 20	
Thfact = 0.2 (see (5.2))	
Distfact = 0.75	
OptQuest search type = *boundary*	
Boundary search parameter = 0.5	
OptQuest Variable Precision = 1.e-4	
Check for duplicates in database = yes	

As discussed in Section 6, OptQuest can insure that all trial points satisfy any linear constraints, and we use this option in our tests below. The boundary search strategy is the OptQuest default, as is its parameter value of

As discussed in Section 6, OptQuest can insure that all trial points satisfy any linear constraints, and we use this option in our tests below. The boundary search strategy is the OptQuest default, as is its parameter value of 0.5. This strategy directs the trial points generated towards the boundary of the region defined by the variable bounds and general linear constraints 50% of the time. For the problem series 8_3_x, the largest of the group with 9 of 10 problems having over 100 variables, we used GRG optimality and feasibility tolerances of 1.e-6 because the default values of 1.e-4 led to GRG termination significantly short of local optimality.

Table 2.3, found in the Appendix, contains the results for 120 of the 131 problems with no discrete variables, sorted by increasing number of variables, with averages for six groups of problems. We exclude the 8_6_1 and 8_6_2 series, which are described separately below. We also exclude eleven problems which were either extremely large, or for which GRG could not find local solutions from most or all starting points. This was almost always due to failure to find a feasible solution, due to termination at a local minimum of the phase one objective. These computations used a 1.3 ghz Dell Optiplex GX400 PC with the Windows 2000 OS. The "fcn call" columns record the total number of times all problem functions are evaluated. The column headed "% gap" is the percentage difference between the best feasible objective value found by OQNLP and the best known value, i.e., the ratio gap = 100*(OQNLP obj – bestobj)/(1+abs(bestobj)).

For minimization problems, and its negative for maximization. Hence a negative *gap* indicates that OQNLP found a feasible point with better objective value than the "best known" value provided in Floudas, et al. (1999). All but 6 of the 120 problems have gaps less than 1%, most much smaller. These 6 are solved to very small gaps using 5000 iterations

and, for 2 problems, increasing the boundary parameter, as is discussed shortly. There are 2 problems with sizeable negative gaps, indicating that OQNLP found a better value than the best reported. The column headed "max abs x" is the largest absolute component of the decision vector, *x,* in the best solution found. Large values indicate a more difficult problem, in the sense that the rectangle defined by the variable bounds is larger, so the search must cover a larger volume.

This "base case" OQNLP algorithm finds its best solution very quickly. The best OQNLP solution is found by the first GRG call in 88 of the 120 problems, and in the second GRG call in 7 more, confirming the effectiveness of stage 1 of the algorithm in finding an initial GRG starting point within the basin of attraction of the global optimum. This happens most often in the smaller problems, but occurs 11 times in the 20 largest.

Table 2.4 aggregates the averages for the six groups of problems in Table 2.3, and includes the following ratios: GRG ratio = average GRG calls to best/average total GRG calls with similar definitions for iterations, function calls, and computation time.

The groups are ordered in terms of increasing number of variables, and the number of local optima found increases with problem size, with an average of 22.6 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 (110 to 141 variables), reflecting the GRG effort required to solve these problems, which have many nonlinear constraints.

Table 2.4. Average performance statistics for 6 groups of problems

VAR RANGE	NO. OF PROBS	ITNS TO BEST	GRG CALLS TO BEST	TOTAL GRG CALLS	GRG RATIO	LOCALS FOUND	FCN CALLS TO BEST	TOTAL FCN CALLS	FCN RATIO	TIME TO BEST	TOTAL TIME	TIME RATIO
1 to 4	31	213.9	1.3	9.2	0.14	2.3	281.7	1582.8	0.18	0.5	2.4	0.21
5 to 7	31	212.5	1.3	11.0	0.12	3.0	361.0	3827.5	0.09	0.3	1.1	0.28
8 to 12	21	293.3	3.2	17.2	0.19	6.1	841.5	3416.6	0.25	0.5	1.5	0.35
14 to 21	17	212.8	1.8	25.6	0.07	6.8	446.8	5030.7	0.09	1.7	4.4	0.38
22 to 78	11	310.2	7.3	34.0	0.21	12.4	1423.4	10528.3	0.14	1.5	4.4	0.33
110 to 141	9	392.4	9.3	26.7	0.35	22.6	20654.7	55580.3	0.37	23.6	54.4	0.43
Overall	**120**	**272.5**	**4.0**	**20.6**	**0.18**	**8.8**	**4001.5**	**13327.7**	**0.19**	**4.7**	**11.4**	**0.33**

Average total GRG calls are fairly stable at between 17 to 34 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 Section 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 0.07 to 0.35, meaning that the best solution is found in the first 7% to 35% 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 OQNLP 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 before the best solution was found.

Table 2.5 shows the results of using 5000 iterations to solve the 6 problems whose gaps in Table 2.3 are greater than 1%.

All problems are solved to within very small gaps. To achieve an essentially zero gap for problems 2_1_6 and 2_1_7_5, we had to change the OptQuest boundary strategy parameter from its default value of 0.5 to 1.0. This causes a strategy that drives trial points towards the boundary of the feasible region defined by the bounds and linear constraints to be used 100% of the time, rather than 50% (see the OptQuest User guide, page 32). These two problems are quadratic programs with concave objectives (to be minimized), so all locally optimal solutions are at extreme points of the feasible region. One would expect a strategy that generates points near the boundary of the feasible region 100% of the time to be most effective on such problems.

Table 2.5. Solving previously unsolved problems in 5000 iterations

PROB	VARS	ITNS TO BEST	GRG CALLS TO BEST	TOTAL GRG CALLS	LOCALS FOUND	FCN CALLS TO BEST	TOTAL FCN CALLS	TIME TO BEST	TOTAL TIME	GAP	BNDY PARAM
EX8_3_7	126	308	12	164	92	7125	505273	15.4	436.3	0.0009	
EX2_1_1	5	893	2	14	12	896	5055	0.12	0.85	0.0000	
EX2_1_6	11	253	2	8	4	256	5042	0.73	8.17	15.0000	0.5
EX2_1_6	11	2591	18	27	12	2703	5171	5.64	8.92	0.0000	1
EX2_1_8	24	2318	32	44	11	3411	6270	6.8	16.91	0.0000	
EX2_1_7_5	21	362	9	36	22	975	7360	3.7	37.7	1.0866	0.5
EX2_1_7_5	21	520	23	90	15	2397	11341	7.18	52.23	0.0002	1
EX14_1_7	11	299	9	220	64	6518	119049	0.51	11.1	0.0000	

Solving Problems with Discrete Variables

Table 2.6 contains results of solving the 11 problems in the Floudas test set which have discrete variables, using 1000 total and 200 stage one iterations. The problems are sorted first by number of discrete variables, then by number of all variables.

Table 2.6. Solution statistics for 13 problems with discrete variables

PROB	VARS	DISC VARS	LIN CONST	NLIN CONST	ITNS TO BEST	GRG CALLS TO BEST	TOTAL GRG CALLS	TOTAL ENUM	LOCAL FOUND	FCN CALLS TO BEST	TOTAL FCN CALLS	TIME TO BEST	TOTAL TIME	GAP, %
EX12_2_2	4	1	2	1	201	1	1	2	1	209	1008	0.53	2.04	0.0000
EX12_2_1	6	3	3	2	249	6	18	8	7	296	1164	0.54	1.25	0.0000
EX12_2_6	6	3	4	1	201	1	7	6	2	202	1045	0.31	0.4	0.0000
EX9_2_9	13	3	11	1	201	1	3	8	3	202	1010	0.42	1.73	0.0000
EX12_2_3_N	8	4	5	4	441	17	20	16	10	1004	1667	0.85	1.25	0.0000
EX12_2_3	12	4	9	4	390	11	20	16	10	1016	1761	1.11	1.96	0.0000
EX9_1_9	18	5	16	1	201	1	21	32	18	202	1069	2.41	8.01	0.0024
EX12_2_5	9	6	9	1	201	1	6	25	1	202	1136	0.91	2.65	0.0000
EX9_1_6	21	6	19	1	201	1	32	64	27	202	1097	1.53	6.73	0.0000
EX9_1_7	24	6	21	1	204	4	41	64	31	256	1328	4	9.64	0.0000
EX9_1_3	30	6	27	1	207	2	42	64	23	214	1212	2.49	11.1	0.0000
EX12_2_4	12	8	4	3	924	78	88	256	16	2145	2501	3.67	4.06	5.4320
EX12_2_4N	12	8	4	3	207	3	40	256	13	218	1191	0.87	2.96	7.1918
averages	13.5	4.8	10.3	1.8	294.5	9.8	26.1	26.1	12.5	489.8	1322	1.5	4.1	1.0

All problems are solved to very small gaps except 12_2_4 and its reformulation, 12_2_4N, which have the same optimal solutions, and have final gaps of 5.4% and 7.2% respectively. Increasing the number of iterations to 5000 or 10,000 does not yield better solutions for these 2 problems. For the other 11 problems, 7 are solved on the first or second GRG call. 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 OQNLP is larger than this value in 3 of the 13 problems, and the two averages are about the same. 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 9.8 versus 26.1 for complete enumeration. As with the continuous variable problems, the best solutions are found in roughly the first 30% of the 1000 iterations on average.

Clearly, the number of discrete variables in these problems is too small to infer whether or not this "base-case" OQNLP algorithm will be competitive with alternative MINLP solvers like DICOPT (interfaced to GAMS) or branch-and-bound (Biegler, et al., 1997, Floudas, 1995). We believe that OQNLP performance with discrete variables can be significantly enhanced by sending information on GRG solutions back to OptQuest. For example, an option that begins by calling GRG to solve a relaxed MINLP (with all discrete variables allowed to be continuous), could terminate immediately if all discrete variables had discrete values in the GRG solution. Otherwise, the discrete variables could be rounded,

and the resulting high quality solution could be returned to OptQuest, influencing the generation of successor trial points.

In 12_2_3 and 12_2_4 the discrete variables appear linearly. (This is required by the widely used DICOPT MINLP solver.) Since OQNLP allows discrete variables to appear nonlinearly, we reformulated these problems into 12_2_3N and 12_2_4N, respectively, where the discrete variables appear nonlinearly. The resulting models have the advantage that when the discrete variables are fixed for the NLP solver, the continuous variables appear linearly. That is why all measures of computational effort are much smaller for the reformulated versions.

For comparison purposes we ran the 11 MINLP problems from the Floudas problem set using DICOPT, with CONOPT2 as the NLP solver and CPLEX as the MILP solver. It solves a NLP and a MILP at each major iteration. The only changes to the models were slight adjustments to lower bounds on some variables to avoid numerical problems encountered otherwise. The statistics are shown in Table 2.7 below, with some OQNLP data repeated for easier comparison. DICOPT solved all but one of the problems to the best-known solution. In the case of EX_12_2_1 the DICOPT NLP Solver was unable to find a feasible solution for the relaxed NLP, and the problem was incorrectly diagnosed as infeasible. Five of the problems are MILP's, and DICOPT simply invokes CPLEX to solve them. For the other five problems, DICOPT found the optimum in 2 or 3 major iterations. Its runtimes are much shorter than OQNLP, and the number of NLP solver calls is much less. Termination was caused by the NLP solver objective worsening, an infeasible MILP, and the relaxed NLP having an integer solution.

It is difficult to infer much from such small problems. However, we expect that DICOPT will be much faster than OQNLP when it succeeds. As a primal method, OQNLP has the potential to find a good solution in cases where DICOPT fails to find a feasible integer solution, and it may sometimes be useful to study the multiple integer feasible solutions that OQNLP can provide.

Varying the Length of Stage One

We have solved the 120 Floudas problems with no discrete variables with three values for the number of stage one iterations: 200 as described above, 300, and as many as are required to generate all "first generation" trial points (those created from the initial population), called the "1gen" strategy. With 1gen and 200 stage one iterations, there were 1000 total iterations, while with 300 we used 1100 total, in order to provide at least 800 iterations in stage 2 for all strategies. The averages for various measures of computational effort and achievement over all 120 problems for these three stage one strategies are shown in Table 2.8 below. The column headed "probs < 1%", shows the number of problems solved to a gap of 1% or less by the strategy, while the last column gives the number

of these successful runs where the best solution was found at the first or second GRG call.

Table 2.7. OQNLP and DICOPT results for MINLP problems

PROB	OQNLP: GRG CALLS	OQNLP: TOTAL RUNTIME	OQNLP: TIME TO BEST SOL.	OQNLP: % GAP TP BEST KNOWN SOL.	DICOPT ITNS	DICOPT TERMINATION	DICOPT: RUNTIME	DICOPT: % GAP TP BEST KNOWN SOL.
EX12_2_2	1	2.04	0.53	0.00	2	infes mip	2.26	0.00
EX12_2_1	18	1.25	0.54	0.00	F	nlp 1 infes	0.06	F
EX12_2_6	7	0.4	0.31	0.00	0	Relaxed nlp integer	0.05	0.00
EX9_2_9	3	1.73	0.42	0.00	0	milp	0.05	0.00
EX12_2_3	20	1.96	1.11	0.00	3	worsen	0.48	0.00
EX9_1_9	21	8.01	2.41	0.00	0	milp	0.11	0.00
EX12_2_5	6	2.65	0.91	0.00	3	worsen	0.4	0.00
EX9_1_6	32	6.73	1.53	0.00	0	milp	0.11	0.00
EX9_1_7	41	9.64	4	0.00	0	milp	0.28	0.00
EX9_1_3	42	11.1	2.49	0.00	0	milp	0.11	0.00
EX12_2_4	88	4.06	3.67	5.43	3	worsen	0.45	0.00

Table 2.8. Effects of varying the number of stage one iterations

STAGE 1 ITNS	ITNS TO BEST	GRG CALLS TO BEST	TOTAL GRG CALLS	LOCALS FOUND	FCN CALLS TO BEST	TOTAL FCN CALLS	TIME TO BEST	TOTAL TIME	PROBS <1%	PROBS IN 1 OR 2 GRG CALLS
163.4	235.6	5.1	27.4	12.6	2127.2	13263.2	5.0	15.9	114	(78,11)
200	272.5	4.0	20.6	8.8	4001.5	13327.7	4.7	11.4	114	(87,7)
300	392.4	5.0	23.0	12.1	6549.9	17672.4	8.1	16.4	109	(90,2)

Examining this table, we see that the iterations and function calls to find the best solution increase with the number of stage one iterations. This is as expected, since the first GRG call comes at the end of stage one, whose purpose is to provide a high quality starting point for GRG.

However, several other effort measures show a minimum at 200 initial iterations: grg calls to best, total grg calls, time to best, and total time. The number of problems solved in one or two GRG calls is also maximized for 200 initial iterations, although the differences between the three strategies are small. Since both the "1gen" and "200" strategies have the same number of problems solved to within a 1% gap, these results imply a mild preference for the (200,1000) strategy.

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 one 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, as shown in Table 2.8. The number of GRG calls is also influenced by other factors, so the effect is not monotonic. We believe that the superior performance of the (200,1000) strategy is due to its achieving a best balance between these competing effects.

8. Minimizing the Potential Energy of a Cluster of Particles

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 decision variables are the x, y, and z components of each particle. For the Lennard-Jones family of potential energy minimization problems 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. Nonlinear constraints are included to avoid objective function singularities where the distance between one or more pairs of points is very small. If they are not included, GAMS encounters many thousands of domain violations-these still occur in the above formulation, but are less frequent. 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 uses the Morse potential, where the Euclidean distance appears in the argument of an exponential function. There is no need to protect against domain violations here, so there are no

constraints except for upper and lower bounds of 5 and –5 on the coordinates of each particle, as in the Lennard-Jones case. 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 they are a rigorous test for global optimization algorithms.

Results of applying OQNLP to these two problem classes using 200 stage one and 1000 total iterations for several values of N are shown in Tables 2.9 and 2.10 below.

Table 2.9. Minimizing the Lennard-Jones potential function

N	VARS	NLIN CONST	ITNS TO BEST	GRG TO BEST	TOTAL GRG	GRG RATIO	LOCALS FOUND	FNC TO BEST	TOTAL FCN	FCN RATIO	TIME TO BEST	TOTAL TIME	TIME RATIO	GAP, %
5	9	10	201	1	16	0.06	8	447	6669	0.07	0.1	1.21	0.08	0.00
10	24	45	437	7	15	0.47	14	6296	13220	0.48	5.07	10.32	0.49	0.00
15	39	105	252	4	17	0.24	17	4775	20641	0.23	9.07	37.39	0.24	0.00
20	54	190	476	11	21	0.52	21	11607	20402	0.57	39.49	69.25	0.57	0.00
25	69	300	478	9	58	0.16	58	13900	109183	0.13	74.87	570.2	0.13	2.71
30	84	435	730	27	48	0.56	48	51221	85873	0.60	399.17	668.8	0.60	1.71
avg	46.5	180.8	429	9.8	29.2	0.3	27.7	14707.7	42664.7	0.3	88.0	226.2	0.4	0.7

Table 2.10. Minimizing the Morse potential function

N	VARS	NLIN CONST	ITNS TO BEST	GRG TO BEST	TOTAL GRG	GRG RATIO	LOCALS FOUND	FNC TO BEST	TOTAL FCN	FCN RATIO	TIME TO BEST	TOTAL TIME	TIME RATIO	GAP, %
5	9	0	201	1	12	0.08	9	328	2666	0.12	0.05	0.31	0.16	0.00
10	24	0	201	1	7	0.14	7	663	3420	0.19	0.37	1.66	0.22	0.00
15	39	0	201	1	14	0.07	14	687	7826	0.09	0.9	8.32	0.11	0.00
20	54	0	284	4	23	0.17	23	1593	10427	0.15	3.42	20	0.17	0.00
25	69	0	300	3	59	0.05	59	2125	31294	0.07	7.2	88.67	0.08	0.00
30	84	0	268	4	41	0.10	41	3196	28385	0.11	15.23	117.2	0.13	0.00
40	114	0	476	12	51	0.24	51	9848	41782	0.24	76.89	306	0.25	0.00
50	144	0	262	4	11	0.36	11	4233	11824	0.36	56.39	151.6	0.37	0.14
avg			274.13	3.75	27.25	0.15	26.88	2834.13	17203	0.17	20.06	86.72	0.19	0.02

OQNLP finds the Morse potential easier to minimize, and solves the 7 smallest instances to essentially zero gaps and the N=50 case to a .14% gap. The 4 smallest instances of the Lennard-Jones problems are also solved to near-zero gaps, but the two largest have gaps of 2.7% and 1.7% respectively. We attribute this partially to the occurrence of many domain violations during the GRG runs. The N=25 run had 93,926

divides by zero and 6717 integer power overflows, while the corresponding figures for N=30 were 133,490 and 9551.

The computational effort needed to achieve these excellent 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 generally less than 0.3 for the Morse potential, but occasionally above 0.5 for the Lennard-Jones. Both function calls and GRG calls to achieve the best solution are quite small, and, for the Morse function, they do not increase rapidly with N. The number of local minima found increases rapidly with N, and is around 60 for N=30 or above. 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 two Lennard-Jones problems with gaps larger than 1% using 200 stage one and 5000 total iterations are shown in Table 2.11 below

Table 2.11. Solving with 5000 iterations

N	VARS	NLIN CONST	ITNS TO BEST	GRG TO BEST	TOTAL GRG	GRG RATIO	LOCALS FOUND	FCN TO BEST	TOTAL FCN	FCN RATIO	TIME TO BEST	TOTAL TIME	TIME RATIO	GAP, %	BND
25	69	300	4089	339	436	0.78	436	664754	820883	0.81	3472.6	4290	0.81	0.00	5
30	84	435	240	5	33	0.15	33	9856	69471	0.14	80.11	554.8	0.14	3.18	5
30	84	435	2486	162	414	0.39	414	188479	432148	0.44	1496	3405	0.44	0.67	3

The N=25 problem is solved to a near-zero gap, but the gap for the N=30 problem (row 2 of the table) actually increases from 1.71% with 1000 iterations to 3.18% with 5000. This is because some aspects of the OptQuest solution strategy depend on the iteration limit, so the two runs use a different sequence of trial points in their first 1000 iterations. The search is more aggressive when there are only 1000 iterations allowed, and this aggressiveness leads to a better final solution in the shorter run. However, if the parameter bnd (each variable has bounds of (-bnd,bnd)) is decreased from 5 to 3, the gap for 5000 iterations decreases to 0.67%, showing the benefits of searching within a smaller rectangle. The computational effort to achieve these improved outcomes, compared to the shorter runs, increases roughly by factors of 5 to 8.

9. Comparison with Random Starts

OptQuest was chosen as the provider of starting points because we felt it would find good points quickly. As a first step to investigating this, we selected the Morse and Lenard_jones potential functions described in section 8, generated either 100 or 200 independent uniformly distributed starting points, started CONOPT2 from each of these, and observed how

many calls found the best known solution, as well as how many distinct local solutions were found. This was done with a LOOP statement in the GAMS models discussed in section 8, and compared with the number of OQNLP calls required before CONOPT2 found the best solution. The results provide a crude estimate of the relative volume of the basin of attraction of the global optimum, as the ratio f=nglob/ncalls, where nglob is the number of NLP solver calls leading to the global solution. The expected number of calls before the global solution is first located is simply 1/f. Results are shown in Tables 2.12 and 2.13 below. Since CONOPT2 is used rather than LSGRG2, the number of OQNLP calls to find the best solution differ from those in Tables 2.9 and 2.10.

Table 2.12. Morse potential function, random starts

ATOMS	VARS	CALLS	NGLOB	DISTINCT LOCALS	EXP CALLS TO BEST	OQNLP CALLS TO BEST
5	9	200	13	17	15.4	1
10	24	200	1	107	200	1
15	39	200	9	161	22.2	1
20	54	200	10	189	20	2
25	69	200	2	185	100	4
30	84	200	2	188	100	17
40	114	200	3	191	66.7	7
50	144	200	3	181	66.7	20

Table 2.13. Lennard-Jones potential, random starts

ATOMS	VARS	CALLS	NGLOB	DISTINCT LOCALS	EXP CALLS TO BEST	OQNLP CALLS TO BEST
5	9	100	99	2	1.0	1
10	24	100	4	27	25.0	21
15	39	100	3	85	33.3	6
20	54	200	2	167	100	67

For both problems, OQNLP finds the best solution in fewer solver calls than the expected number of calls to the best solution for random starts, much fewer for the Morse potential function. For both problem sets, the relative volume estimate *f* decreases quickly as problem size increases, and many more than 200 solver calls are needed to estimate it accurately. For the larger numbers of atoms, almost every solver call leads to a different local minimum.

10. Summary and Future Research

While the performance of this "base case" OQNLP algorithm on problems with only continuous variables is quite good, there are options which promise improvements. OptQuest's search is usually more efficient when the initial population contains high quality solutions. No such solutions are supplied in the computational experiments described here. A way to provide one good solution is to call GRG at the start of stage one, communicating the local optimum found to OptQuest as a possible member of its initial population. GRG's starting point could be either user-provided, or the best point found in some initial set of OptQuest iterations (perhaps a few hundred as in the current stage one). The latter option is equivalent to adding a new stage one consisting of these initial OptQuest iterations, calling GRG from the best stage one solution (stage 2), and doing a stage 3 by placing the GRG solution as a candidate point in a newly initialized population, before performing another set of OptQuest iterations. In our computational experiments, the GRG solution resulting from a start at the best point from 200 or so OptQuest iterations is globally optimal in about 75% of the problems solved. Hence the initial stage 3 population would often contain the global optimum, plus points diverse from it. The final stage (4) would be the current stage 2, where GRG is called repeatedly at points which are accepted by the distance and merit filters. We are currently implementing this option.

The above idea is naturally extended by communicating other GRG solutions to OptQuest during (the current) stage 2, but this must be done in a way that maintains enough diversity in the population. Hence only unique local solutions should be sent to OptQuest, and these should not be too close to one another. Some distance threshold must be devised, and OptQuest would receive only local solutions whose distance from the nearest previously found local solution is greater than the threshold.

In problems with discrete variables, high quality points for OptQuest's initial population can be determined by solving the relaxed MINLP, where all discrete variables are allowed to be continuous within their bounds. If all discrete variables take on allowed values, this solution is at least locally optimal. If not, various rounding procedures can be applied to it to generate one or more high quality discrete solutions. We are currently implementing this option as well.

Another promising option for MINLP's is to "hide" the continuous variables from OptQuest, which searches only over the space of discrete variables. It is aware only of the constraints involving only discrete variables. This allows it to focus its attention on these key variables, which GRG cannot vary. That is, OptQuest is applied to the projection of the problem (1.1)-(1.4) onto y-space. This projected problem is to minimize

$$F(y) = \min_x (f(x,y) \mid gl \le G(x,y) \le gu, l \le A_1 x + A_2 y \le u, x \in S)$$

over all constraints involving only *y*. GRG is then applied to the *x* sub-problem on the right hand side of the above equation. As is done currently, GRG would fix the discrete variables at values specified by OptQuest, and optimize over the continuous variables. If GRG finds a feasible solution, its optimal objective value is returned to OptQuest. If not, the exact penalty function value of this solution is returned, using some set of sufficiently large penalty weights. These GRG solutions should be of much higher quality than the continuous variable values generated by OptQuest in the current algorithm, where the continuous variables are nowhere near locally optimal for the associated discrete variables.

Comparative tests of OQNLP and alternative global optimization methods are also needed. GAMS Development Company has interfaced several global and MINLP solvers, including OQNLP, to GAMS, and this will make the comparison process much easier by providing a common computing environment and model base. The model base has been expanded by a website recently introduced by GAMS Development Company called "GAMS World" at www.gamsworld.org (see ad on the back cover of ORMS Today, Aug 2001). This is divided into "MINLP world" and "Global world". MINLP world currently contains a set of MINLP test problems coded in GAMS, facilities for converting models in several other algebraic modeling languages to GAMS models, and other information on MINLP. Global world contains similar information for global optimization. We also plan comparative tests with the global optimizers in Frontline Systems Premium Excel Solver.

Appendix

Detailed computational results for 120 Floudas test problems.

Table 2.3. Results for Floudas GAMS problems with no discrete variables

PROBLEM NAME	VARS	LIN CONST	NONLIN CONST	INIT ITNS	ITNS TO BEST	GRG CALLS TO BEST	TOTAL GRG CALLS	LOCALS FOUND	FCN CALLS TO BEST	TOTAL FCN CALLS	TIME TO BEST (SEC)	TOTAL TIME (SEC)	MAX ABS X COMP	GAP, %
EX4_1_1	1	0	0	200	201	1	5	1	211	1139	0.17	0.61	1.19	0.000
EX4_1_2	1	0	0	200	201	1	6	1	211	1208	0.22	0.99	1.09	0.000
EX4_1_3	1	0	0	200	201	1	4	1	212	1068	0.17	0.5	6.33	0.000
EX4_1_4	1	0	0	200	201	1	8	2	202	1090	0.16	0.49	2	0.000
EX4_1_6	1	0	0	200	201	1	7	2	212	1119	0.17	0.55	3	0.000
EX4_1_7	1	0	0	200	201	1	12	1	207	1190	0.16	0.71	1	0.000

Table 2.3. Results for Floudas GAMS problems with no discrete variables (cont'd)

PROBLEM NAME	VARS	LIN CONST	NONLIN CONST	INIT ITNS	ITNS TO BEST	GRG CALLS TO BEST	TOTAL GRG CALLS	LOCALS FOUND	FCN CALLS TO BEST	TOTAL FCN CALLS	TIME TO BEST (SEC)	TOTAL TIME (SEC)	MAX ABS X COMP	GAP, %
EX8_1_2	1	0	0	200	201	1	15	3	208	1161	0.16	0.71	5.17	0.010
EX14_1_9	2	0	2	200	201	1	3	2	215	1069	0.22	0.6	347.31	0.000
EX4_1_5	2	0	0	200	201	1	8	2	202	1113	0.17	0.61	1.75	0.000
EX4_1_8	2	0	1	200	201	1	3	3	839	1392	0.66	0.94	3	-0.001
EX4_1_9	2	0	2	200	251	4	9	4	331	1186	0.28	0.66	3.44	-0.001
EX8_1_1	2	0	0	200	201	1	6	2	210	1110	0.16	0.6	2	0.000
EX8_1_3	2	0	0	200	201	1	2	1	226	1054	0.17	0.44	1	0.000
EX8_1_4	2	0	0	200	201	1	11	1	202	1154	0.16	0.66	0	0.000
EX8_1_5	2	0	0	200	201	1	6	2	218	1134	0.17	0.61	0.71	0.000
EX8_1_6	2	0	0	200	205	2	6	3	231	1098	0.22	0.55	8	0.000
EX14_1_1	3	0	4	200	201	1	7	4	223	1367	0.28	0.93	3.39	0.003
EX14_1_3	3	0	4	200	201	1	12	6	216	2016	0.17	1.6	6.94	0.002
EX14_1_4	3	0	4	200	201	1	9	4	235	1633	0.27	1.26	3.14	0.000
EX3_1_4	3	2	1	200	201	1	13	3	202	1329	0.39	1.93	3	0.000
EX6_2_11	3	1	0	200	201	1	9	2	229	1529	1.15	5.66	0.99	0.000
EX6_2_6	3	1	0	200	201	1	10	2	211	1379	1.1	5.11	0.94	0.000
EX6_2_8	3	1	0	200	201	1	23	3	235	1893	1.26	6.37	0.97	-0.001

Table 2.3. Results for Floudas GAMS problems with no discrete variables (cont'd)

PROBLEM NAME	VARS	LIN CONST	NONLIN CONST	INIT ITNS	ITNS TO BEST	GRG CALLS TO BEST	TOTAL GRG CALLS	LOCALS FOUND	FCN CALLS TO BEST	TOTAL FCN CALLS	TIME TO BEST (SEC)	TOTAL TIME (SEC)	MAX ABS X COMP	GAP, %
EX14_1_8_N	3	0	0	200	548	6	9	2	726	1256	0.54	0.98	0.65	0.013
EX6_1_2	4	1	2	200	201	1	8	2	233	1890	0.82	4.06	1	0.006
EX6_2_12	4	2	0	200	201	1	14	4	232	1716	0.99	5.06	0.5	0.000
EX6_2_14	4	2	0	200	201	1	17	2	207	1221	0.66	3.63	0.5	-57.797
EX6_2_9	4	2	0	200	201	1	23	2	260	2885	1.05	7.03	0.5	0.000
EX7_3_1	4	6	1	200	201	1	10	1	805	8361	1.31	11.59	1073.39	0.003
EX7_3_2	4	6	1	200	201	1	4	1	381	1288	1.53	5.43	1.28	0.000
EX9_2_8	4	3	2	200	201	1	5	1	202	1020	0.44	2.64	1	0.000
averages	**2.5**	**0.9**	**0.8**	**200.0**	**213.9**	**1.3**	**9.2**	**2.3**	**281.7**	**1582.8**	**0.5**	**2.4**	**47.9**	**-1.9**
EX14_2_1	5	1	6	200	201	1	13	1	247	1812	0.34	1.11	54.25	0.000
EX14_2_4	5	1	6	200	201	1	13	1	271	1990	0.38	1.22	72.97	0.000
EX14_2_6	5	1	6	200	201	1	9	1	245	1549	0.38	1.16	61.59	0.000
EX14_2_8	5	1	4	200	201	1	1	1	230	1029	0.29	0.78	55.73	0.000
EX2_1_1	5	1	0	200	201	1	4	4	202	1007	0.05	0.16	1	2.778
EX3_1_2	5	0	6	200	201	1	8	1	235	1364	0.04	0.14	78	-0.002
EX7_3_3	5	6	2	200	201	1	7	2	254	11992	0.36	2	2.81	0.002
EX8_1_7	5	0	5	200	291	3	7	3	905	3191	0.06	0.16	2.84	0.001
EX8_5_3	5	3	2	200	201	1	8	3	223	1393	0.39	1.24	0.98	-0.013
EX8_5_4	5	3	2	200	201	1	12	4	303	2004	0.43	1.53	0.78	-0.006
EX8_5_5	5	3	2	200	346	4	10	3	692	2066	0.53	1.41	0.8	-0.345
EX14_1_2	6	0	9	200	201	1	20	1	480	6125	0.05	0.33	31.33	0.000
EX14_1_5	6	4	2	200	201	1	5	2	202	1084	0.47	1.48	1.42	0.000
EX14_2_2	6	1	4	200	201	1	9	1	222	1207	0.34	0.86	58.13	0.000
EX14_2_5	6	1	4	200	201	1	12	1	225	1320	0.37	1	77.19	0.000
EX14_2_7	6	1	8	200	201	1	5	1	233	1229	0.61	2.02	63.56	0.000

Table 2.3. Results for Floudas GAMS problems with no discrete variables (cont'd)

PROBLEM NAME	VARS	LIN CONST	NONLIN CONST	INIT ITNS	ITNS TO BEST	GRG CALLS TO BEST	TOTAL GRG CALLS	LOCALS FOUND	FCN CALLS TO BEST	TOTAL FCN CALLS	TIME TO BEST (SEC)	TOTAL TIME (SEC)	MAX ABS X COMP	GAP, %
EX14_2_9	6	1	4	200	201	1	12	1	219	1253	0.36	1.08	60.63	0.000
EX2_1_2	6	2	0	200	201	1	16	1	202	1246	0.16	0.63	20	0.000
EX2_1_4	6	5	0	200	201	1	8	1	208	1116	0.35	0.76	6	0.000
EX3_1_3	6	4	2	200	201	1	9	5	202	1062	0.18	0.5	10	0.000
EX6_1_4	6	1	3	200	201	1	4	2	546	1974	0.45	1.25	1	-0.004
EX6_2_10	6	3	0	200	201	1	6	1	242	1426	0.46	2.01	0.4	-0.557
EX6_2_13	6	3	0	200	201	1	16	13	202	1695	0.45	2.28	0.62	-0.001
EX7_2_2	6	0	5	200	201	1	9	6	242	1458	0.04	0.14	11.02	0.000
EX8_1_8	6	0	5	200	201	1	9	6	242	1458	0.04	0.14	11.02	-0.001
EX8_5_1	6	3	2	200	224	3	13	11	559	3437	0.46	1.77	0.83	0.931
EX8_5_2	6	2	2	200	282	2	3	2	591	1860	0.47	1.75	0.69	-0.123
EX8_5_6	6	2	2	200	201	1	8	1	283	1661	0.48	1.64	0.67	0.003
EX14_2_3	7	1	8	200	201	1	3	1	240	1150	0.5	1.58	57.16	0.000
EX5_2_4	7	3	3	200	201	1	50	1	335	7541	0.24	0.98	100	0.000
EX7_2_1	7`	2	12	200	218	3	31	10	1709	49954	0.32	2.29	3031.6	-2.176
averages	**5.7**	**1.9**	**3.7**	**200.0**	**212.5**	**1.3**	**11.0**	**3.0**	**361.0**	**3827.5**	**0.3**	**1.1**	**125.0**	**0.0**
EX3_1_1	8	3	3	200	201	1	11	1	1084	8837	0.29	0.74	5109.9	-0.001
EX5_4_2	8	3	3	200	201	1	11	1	539	4761	0.24	0.75	5485.3	0.000
EX6_1_1	8	2	4	200	872	12	14	10	1622	1762	0.66	0.76	1	0.246
EX7_2_3	8	3	3	200	291	5	5	3	2388	3097	0.33	0.8	5110.2	0.003
EX7_2_4	8	0	4	200	657	6	12	6	3586	8191	0.17	0.33	9.81	-0.673
EX9_2_4	8	5	2	200	201	1	34	1	210	1357	0.32	1.28	3	0.000
EX9_2_5	8	4	3	200	201	1	2	1	211	1019	0.55	1.64	7	0.000
EX14_1_6	9	1	14	200	201	1	13	4	290	2822	0.57	1.96	1	0.000
EX6_2_5	9	3	0	200	519	16	22	20	1961	3205	1.52	2.65	31.46	-0.003
EX6_2_7	9	3	0	200	201	1	14	3	272	2373	0.51	2.25	0.45	-0.004
EX14_1_7	11	0	17	200	209	2	22	13	1099	10871	0.14	1.02	10	13.499
EX2_1_5	10	11	0	200	201	1	85	1	266	5570	0.84	3.4	1	0.002
EX2_1_6	11	0	5	200	253	2	3	2	256	1007	0.77	1.85	1	15.000
EX2_1_9	11	1	0	200	201	1	11	3	212	1333	0.31	1.09	0.33	0.000
EX9_1_2	10	5	4	200	201	1	4	1	202	1013	0.37	1.27	4	0.000

Table 2.3. Results for Floudas GAMS problems with no discrete variables (cont'd)

PROBLEM NAME	VARS	LIN CONST	NONLIN CONST	INIT ITNS	ITNS TO BEST	GRG CALLS TO BEST	TOTAL GRG CALLS	LOCALS FOUND	FCN CALLS TO BEST	TOTAL FCN CALLS	TIME TO BEST (SEC)	TOTAL TIME (SEC)	MAX ABS X COMP	GAP, %
EX9_1_4	10	5	4	200	201	1	2	1	202	1009	0.54	1.58	24	0.000
EX9_2_1	10	5	4	200	293	2	17	15	296	1038	0.68	2.02	16.38	0.000
EX9_2_2	10	7	4	200	201	1	31	3	207	1216	0.36	1.36	20	-0.021
EX9_2_7	10	5	4	200	293	2	17	15	296	1038	0.69	1.95	16.38	0.000
EX6_1_3	12	0	9	200	361	8	24	23	1679	4837	1.07	2.26	1	0.354
EX7_3_4	12	0	17	200	201	1	7	1	793	5393	0.04	0.25	731.97	0.000
averages	**9.5**	**3.1**	**5.0**	**200**	**293.3**	**3.2**	**17.2**	**6.1**	**841.5**	**3416.6**	**0.5**	**1.5**	**789.8**	**1.4**
EX2_1_3	14	9	0	200	201	1	13	5	202	1199	0.31	0.9	3	0.000
EX9_1_1	14	7	5	200	201	1	1	1	211	1010	0.72	2.69	14	0.000
EX9_1_5	14	7	5	200	201	1	12	12	202	1023	0.34	1.33	50.59	0.000
EX8_4_6	14	0	8	200	263	3	52	28	3293	44031	0.23	2.17	10	0.001
EX9_1_10	15	7	5	200	201	1	27	9	209	1197	0.32	1.41	100	0.000
EX9_1_8	15	7	5	200	201	1	27	9	209	1197	0.35	1.49	100	-54.545
EX8_4_5	16	0	11	200	201	1	14	2	341	2704	0.1	0.3	0.23	0.000
EX5_4_3	17	9	4	200	340	12	58	2	789	3762	0.77	1.95	310	0.001
EX9_2_3	17	9	6	200	201	1	5	1	202	1051	0.7	2.36	30	0.000
EX9_2_6	17	6	6	200	201	1	8	1	202	1051	0.37	1.28	1	0.000
EX8_4_4	18	0	12	200	201	1	71	1	293	8462	0.07	0.67	5.13	0.000
EX2_1_10	21	10	0	200	201	1	79	3	283	9854	0.73	2.08	66.35	0.000
EX2_1_7_1	21	10	0	200	201	1	10	6	217	1474	4.91	12.62	28.8	0.000
EX2_1_7_2	21	10	0	200	201	1	20	13	217	2245	4.92	10.62	28.8	0.000
EX2_1_7_3	21	10	0	200	201	1	18	10	217	2078	4.84	10.56	28.8	0.000
EX2_1_7_4	21	10	0	200	201	1	12	7	217	1600	4.99	12.43	28.8	0.000
EX2_1_7_5	21	10	0	200	201	1	9	6	291	1584	3.66	9.95	28.8	1.087
averages	**17.5**	**7.1**	**3.9**	**200**	**212.8**	**1.8**	**25.6**	**6.8**	**446.8**	**5030.7**	**1.7**	**4.4**	**49.1**	**-3.1**

Table 2.3. Results for Floudas GAMS problems with no discrete variables (cont'd)

PROBLEM NAME	VARS	LIN CONST	NONLIN CONST	INIT ITNS	ITNS TO BEST	GRG CALLS TO BEST	TOTAL GRG CALLS	LOCALS FOUND	FCN CALLS TO BEST	TOTAL FCN CALLS	TIME TO BEST (SEC)	TOTAL TIME (SEC)	MAX ABS X COMP	GAP, %
EX8_4_1	22	0	10	200	201	1	8	1	257	1774	0.08	0.26	7.5	0.001
EX5_3_2	22	7	9	200	201	1	42	1	471	13484	0.56	1.58	300	0.000
EX2_1_8	24	10	0	200	605	15	20	6	1041	1506	1.6	2.5	24	27.391
EX8_4_2	24	0	10	200	201	1	4	1	300	1400	0.1	0.3	7.45	-0.004
EX5_4_4	27	13	6	200	564	29	75	6	5346	14621	2.15	4.17	200	0.002
EX5_2_5	32	8	11	200	556	21	50	43	4550	15865	2.15	4.21	200	0.000
EX8_4_3	52	0	25	200	201	1	10	1	260	3176	0.36	1.42	4.51	0.000
EX8_2_1	55	6	25	200	222	2	40	7	447	4851	1.44	5.26	236.25	0.001
EX8_2_4	55	6	75	200	201	1	37	7	377	7041	1.74	6.18	216	0.003
EX8_4_7	62	0	40	200	201	1	21	2	340	30603	0.75	5.13	754.96	0.022
EX8_3_9	78	18	27	200	259	7	67	61	2268	21490	5.11	17.32	1000	-0.100
averages	**41.2**	**6.2**	**21.6**	**200.0**	**310.2**	**7.3**	**34.0**	**12.4**	**1423.4**	**10528.3**	**1.5**	**4.4**	**268.2**	**2.5**
EX8_3_1	115	17	59	200	307	10	38	28	67165	152573	30.72	81.99	10000	0.29
EX8_3_2	110	27	49	200	945	15	18	18	12254	16103	33.82	35.21	243.85	0.00
EX8_3_3	110	27	49	200	201	1	19	18	2357	28809	8.65	37.17	223.22	0.00
EX8_3_4	110	27	49	200	203	2	23	22	4875	31568	10.49	39.66	105.54	0.00
EX8_3_5	110	27	49	200	201	1	13	11	2442	12746	10.54	39.59	127.28	0.10
EX8_3_6	110	27	49	200	511	14	37	29	6760	56896	20.21	44.97	1000	0
EX8_3_7	126	27	65	200	201	1	8	7	2868	6401	10.99	32.76	100	9.7517
EX8_3_8	126	28	65	200	414	20	45	42	79983	156082	51.32	101.55	10000	0.0004
EX8_3_10	141	43	65	200	549	20	39	28	7188	39045	35.68	76.39	6000	0.3604
averages	**117.6**	**27.8**	**55.4**	**200.0**	**403.1**	**9.3**	**25.3**	**21.9**	**14840.9**	**43456.3**	**22.7**	**50.9**	**2225.0**	**1.3**

References

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

Brooke, A., D. Kendrick, A. Meeraus and R. Raman (1992) *GAMS: A User's Guide*, Boyd and Fraser, Danvers, MA.

Dixon, L. and 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*, 6(2):207-216.

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

Floudas C.A., P.M. Pardalos, C.S. Adjiman, W.R. Esposito, Z. Gumus, S.T. Harding, J.L. Klepeis, C.A. Meyer and C.A. Schweiger (1999) *Handbook of Test Problems in Local and Global Optimization,* Kluwer Academic Publishers.

Frontline Systems, Inc. (2000) "Premium Solver Platform User Guide," 95-96.

Laguna, M. and R. Martí (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. and R. Martí (2001) "The OptQuest Callable Library," to appear in *Optimization Software Class Libraries,* Stefan Voβ and D. Woodruff, eds., Kluwer Academic Publishers, Boston.

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

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

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

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

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

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

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

Smith, S. and L. Lasdon (1992) "Solving Large Sparse Nonlinear Programs Using GRG," *ORSA Journal on Computing*, 4(1): 3-15.