SOLVING SYSTEMS OF NONLINEAR EQUATIONS WITH CONTINUOUS GRASP

MICHAEL J. HIRSCH, PANOS M. PARDALOS, AND MAURICIO G. C. RESENDE

ABSTRACT. A method for finding all roots of a system of nonlinear equations is described. Our method makes use of C-GRASP, a recently proposed continuous global optimization heuristic. Given a nonlinear system, we solve a corresponding adaptively modified global optimization problem multiple times, each time using C-GRASP, with areas of repulsion around roots that have already been found. The heuristic makes no use of derivative information. We illustrate the approach on systems from the literature.

1. INTRODUCTION

The solution to many physical problems are encapsulated as the roots of a system of nonlinear, many variable, equations [1]. Numerous examples from all branches of the sciences are given in [1, 10, 21, 23]

If the system is linear, then there are many approaches available to determine the solutions (e.g. Gaussian elimination and matrix inverses [11], null-space computation [11, 15], and linear programming [2]). If the system happens to be polynomial in nature, then exact techniques employing resultants [4] or Gröbner bases [5, 6] can be employed. However, these techniques have their shortcomings. With resultants, some ordering of the eliminated variables can lead to extraneous solutions. There is no way to determine whether an ordering will cause such extraneous solutions, nor which solutions might be extraneous. Gröbner bases using Buchberger's algorithm suffer from the generation of a large number of equations during the intermediate stages of computation.

When the equations in the system do not exhibit nice linear or polynomial properties, then it can be very difficult to determine roots of the system. In this general case, we are left with numerical routines to determine roots of the system. Unfortunately, there is no numerical method that can guarantee finding all the roots to a general system of equations in finite time. Most numerical solution techniques are based on Newton's method [1, 18]. Each iteration of Newton's method consists of computing the Jacobian matrix at a feasible point and solving a linear system which approximates the original system at that point. The solution to the linear system is used as the starting point in the next iteration. Another class of numerical methods transforms the system into an optimization problem. In a strategy based on trust regions [18], at each iteration a convex quadratic function is minimized to determine the next feasible point to step to. The convex quadratic function is the squared norm of the original system plus a linear function multiplied by the Jacobian matrix. There is also the approach of homotopy methods, sometimes referred to as continuation methods [18, 22]. This approach begins with a 'starting' system of equations (not the true system)

Date: November 2, 2006.
Key words and phrases. Nonlinear systems of equations, global optimization, continuous optimization, heuristic, stochastic algorithm, stochastic local search, nonlinear programming, GRASP, C-GRASP..
AT& Labs Research Technical Report TD-6V6JCX..

whose solution is known. This starting system is gradually transformed to the original system. At each stage, the current system is solved to find a starting solution for the next stage system. The idea is that as the system changes, the solutions trace out a path from a solution of the starting system to a solution of the original system. At each stage, the current system is normally solved by a Newton-type method, again making use of the Jacobian. Problems with this approach include not all paths converging to a finite solution.

With all of the above numerical methods for solving nonlinear systems of equations, derivative information is assumed to be available. Also, these approaches do not consider the problem of finding all solutions to a nonlinear system, just one solution. In this paper, we address the problem of finding all roots to a system of equations. We assume all roots of the system are real. However a relatively simple modification to our implementation of the heuristic will allow complex roots to be determined also. Given the nonlinear system, we construct a corresponding optimization problem, which we solve multiple times, making use of *continuous GRASP*, or *C-GRASP*, a new metaheuristic for continuous global optimization. The optimization problem solved each time is adaptively modified as roots of the system are discovered.

This paper is organized as follows. In Section 2, we describe the enhanced version of C-GRASP [12, 13]. In Section 3, we briefly discuss transforming the system of equations into a corresponding global optimization problem with the goal of finding all roots of the original system. We apply this approach to several systems from the literature. Conclusions are drawn in Section 4.

2. ENHANCED CONTINUOUS GRASP

Feo and Resende [7] describe the metaheuristic GRASP as a multi-start local search procedure, where each GRASP iteration consists of two phases, a construction phase and a local search phase. In the construction phase, interactions between greediness and randomization generate a diverse set of quality solutions. The local search phase improves upon the solutions found in the construction phase. The best solution over all of the multi-start iterations is retained as the final solution. Festa and Resende [8] give a detailed bibliography of numerous papers devoted to applying GRASP to combinatorial optimization problems. Recently, in [12], the GRASP approach was modified for continuous global optimization problems resulting in C-GRASP. That paper showed the performance of C-GRASP on a set of 14 standard test functions as well as hard real-world optimization problems from the areas of robot kinematics and chemical equilibria. In [14], C-GRASP was shown to perform well when compared to two commonly used heuristics [3, 17] on a problem from network sensor registration. In addition, a follow-on paper to [12] details enhancements to C-GRASP [13]. The enhanced C-GRASP was shown to perform well when compared to an advanced tabu search, a scatter search, and a genetic algorithm on a set of 40 test functions.

Pseudo-code for C-GRASP is shown in Figure 1. For each multi-start iteration, C-GRASP begins with an initial coarse discretization of the search space and an initial random solution vector x. Each iteration proceeds with a call to the construction procedure and the local search procedure. The output from construction is used as the input to local search, and the output from local search is used as the input to construction in the next iteration. As the iterations progress, the discretization adaptively becomes more dense as C-GRASP converges to a minimum. The best solution found over all multi-start iteratioons is returned as the minimum.

```
procedure C-GRASP($n,\ell,u,f(\cdot),h_s,h_e,\rho_{lo}$)
1  $f^* \leftarrow \infty$;
2  while Stopping criteria not met do
3      $x \leftarrow$ UnifRand($\ell,u$);
4      $h \leftarrow h_s$;
5      while $h \geq h_e$ do
6          $\mathtt{Impr_C} \leftarrow$ false;
7          $\mathtt{Impr_L} \leftarrow$ false;
8          $[x,\mathtt{Impr_C}] \leftarrow$ ConstructGreedyRandomized($x,f(\cdot),n,h,\ell,u,\mathtt{Impr_C}$);
9          $[x,\mathtt{Impr_L}] \leftarrow$ LocalImprovement($x,f(\cdot),n,h,\ell,u,\rho_{lo},\mathtt{Impr_L}$);
10         if $f(x) < f^*$ then
11             $x^* \leftarrow x$;
12             $f^* \leftarrow f(x)$;
13         end if
14         if $\mathtt{Impr_C}$ = false and $\mathtt{Impr_L}$ = false then
15             $h \leftarrow h/2$;     /* make grid more dense */
16         end if
17     end while
18 end while
19 return($x^*$);
end C-GRASP;
```

FIGURE 1. Pseudo-code for enhanced C-GRASP.

We briefly describe the construction and local search phases of C-GRASP. Pseudo-code for the construction phase is shown in Figure 2. It takes as input a current solution x. Each of the coordinate directions is marked as *unfixed*. Centered at x, for each coordinate direction i that is unfixed, we perform a line search in coordinate direction i, locating the position (using the current discretization) in direction i where the minimum occurs, say z_i, as well as the value of the minimum, say g_i. A restricted candidate list (RCL) is formed allowing as candidates only those unfixed coordinate directions with g_i values lower than an adaptive threshold. In this way, the RCL contains a list of good candidates. One of the coordinate directions in the RCL is chosen randomly, call it j, and x_j is set equal to z_j, coordinate j is *fixed*, and the process repeats until all coordinates are *fixed*. At this stage, the solution vector x is returned.

Note that, at any time, the best candidate from the RCL is not necessarily the candidate chosen to be *fixed*. Allowing some randomness to decide which of the 'good' coordinates to fix next helps C-GRASP to examine more of the search space, and avoid the trappings of local, but not global, minima. Also, because we are performing the line searches along the coordinate directions, while the output of the construction phase may be a good solution, it will not necessarily be optimal.

Therefore, after construction, we apply a local search procedure that searches in the neighborhood of the current solution for a better solution. The local search procedure is seen in Figure 3. The neighborhood $B_h(x)$ of the current solution x is defined to be the projection of all the current grid points onto the hyper-sphere centered at x and with radius equal to the current discretization level h. When a solution is found that is better than those

```
procedure ConstructGreedyRandomized($x, f(\cdot), n, h, \ell, u$, $\mathtt{Impr_C}$)
    UnFixed ← $\{1,2,\ldots,n\}$;
    $\alpha$ ← UnifRand(0.0, 1.0);
    ReUse ← false;
    while UnFixed $\neq \emptyset$ do
        min ← $+\infty$;
        max ← $-\infty$;
        for $i = 1,\ldots,n$ do
            if $i \in$ UnFixed then
                if ReUse = false then
                    $z_i$ ← LineSearch($x, h, i, n, f(\cdot), \ell, u$);
                    $g_i \leftarrow f(\check{x}^i)$;
                end if
                if min $> g_i$ then min ← $g_i$;
                if max $< g_i$ then max ← $g_i$;
            end if
        end for
        RCL ← $\emptyset$;
        Threshold ← min + $\alpha$ * (max − min);
        for $i = 1,\ldots,n$ do
            if $i \in$ UnFixed and $g_i \leq$ Threshold then
                RCL ← RCL $\cup \{i\}$;
            end if
        end for
        $j$ ← RandomlySelectElement(RCL);
        if $x_j = z_j$ then
            ReUse ← true;
        else
            $x_j \leftarrow z_j$;
            ReUse ← false;
            $\mathtt{Impr_C}$ ← true;
        end if
        UnFixed ← UnFixed $\setminus \{j\}$;      /* Fix coordinate j. */
    end while
    return($x$, $\mathtt{Impr_C}$);
end ConstructGreedyRandomized;
```

FIGURE 2. Pseudo-code for C-GRASP enhanced construction phase.

in its neighborhood, this solution is returned from the local search phase. The interested reader is referred to [12, 13] for details on the C-GRASP heuristic.

3. COMPUTATIONAL EXPERIMENTS

This section is divided into three parts. First, we briefly define the optimization problem considered as a means of finding all roots to a system of equations. Then, we discuss our testing environment. Finally, we apply our approach to some systems from the literature.

```
procedure LocalImprovement($x, f(\cdot), n, h, \ell, u, \rho_{lo}$, Impr_L)
1  $x^* \leftarrow x$;
2  $f^* \leftarrow f(x)$;
3  NumGridPoints $\leftarrow \prod_{i=1}^{n} \lceil (u_i - \ell_i)/h \rceil$;
4  MaxPointsToExamine $\leftarrow \lceil \rho_{lo} *$ NumGridPoints$\rceil$;
5  NumPointsExamined $\leftarrow 0$;
6  while NumPointsExamined $\leq$ MaxPointsToExamine do
7      NumPointsExamined $\leftarrow$ NumPointsExamined $+ 1$;
8      $x \leftarrow$ RandomlySelectElement($B_h(x^*)$);
9      if $\ell \leq x \leq u$ and $f(x) < f^*$ then
10         $x^* \leftarrow x$;
11         $f^* \leftarrow f(x)$;
12         Impr_L $\leftarrow$ true;
13         NumPointsExamined $\leftarrow 0$;
14     end if
15 end while
16 return($x^*$, Impr_L);
end LocalImprovement;
```

FIGURE 3. Pseudo-code for C-GRASP enhanced local improvement phase.

3.1. **Adaptive optimization problem.** Suppose we are given the system of equations $f_1(x), \ldots, f_r(x)$, where $x \in \mathbb{R}^n$ and $f_i : \mathbb{R}^n \longrightarrow \mathbb{R}$ for each $i \in \{1, \ldots, r\}$. A solution to this system is a vector $x^* \in \mathbb{R}^n$ such that $f_i(x^*) = 0$ for each $i \in \{1, \ldots, r\}$. We transform this system into the functional

$$F(x) = \sum_{i=1}^{r} f_i^2(x). \tag{1}$$

It is clear that all solutions to the system $f_1(x), \ldots, f_r(x)$ are precisely those global minimizers of (1) that evaluate $F(x)$ to 0.

We would like to apply C-GRASP to the problem of minimizing (1). However, our goal is not to find one solution to the original system of equations, but to find all solutions. Suppose the original system has K roots. Minimizing (1) K times using C-GRASP (or any heuristic, for that matter) with different starting solutions gives no guarantee of finding all K roots. It is entirely possible that some of the roots would be found multiple times, while other roots would not be found at all. To overcome this problem, we choose to adaptively modify the objective function $F(x)$ as roots are found. Suppose that C-GRASP has just found the k-th root (roots are denoted $x^1, \ldots, x^k$). Then it will restart, with modified objective function given by

$$F(x) = \sum_{i=1}^{r} f_i^2(x) + \beta \sum_{j=1}^{k} e^{-\|x - x^j\|} \chi_\rho(\|x - x^j\|), \tag{2}$$

where

$$\chi_\rho(\delta) = \begin{cases} 1 & \text{if } \delta \leq \rho \\ 0 & \text{otherwise} \end{cases}$$

is the characteristic function [25], β is a large constant, and ρ is a small constant.

These adaptive modifications to $F(x)$ have the effect of creating an area of repulsion (or penalty region) around solutions that have already been found. Note that it is important to choose ρ small enough so that other solutions not yet found are not adversely penalized in the modified objective function.

3.2. **Test environment.** The experiments to follow were carried out on a Dell PowerEdge 2600 computer with dual 3.2 GHz 1 Mb cache XEON III processors and 6 Gb of memory running Red Hat Linux 3.2.3-53. The implementation of C-GRASP was done in the C++ programming language and compiled with GNU `g++` version 3.2.3, using compiler options `-O6 -funroll-all-loops -fomit-frame-pointer -march=pentium4`. The algorithm used for random-number generation is an implementation of the Mersenne Twister algorithm described in [19]. Also, in (2), we set β to 1000 and ρ to 0.01 for each of the problems considered. We consider a root to be found if the objective function value becomes smaller than 10^{-7}.

3.3. **Test systems.** The first application we consider, originally described in [16], but also found in [9] and [10], concerns a model of two continuous non-adiabatic stirred tank reactors. These reactors are in a series, at steady state with a recycle component, and have an exothermic first-order irreversible reaction. When certain variables are eliminated, the model results in a system of two nonlinear equations

$$f_1 = (1-R)\left[\frac{D}{10(1+\beta_1)} - \phi_1\right]\exp\left(\frac{10\phi_1}{1+10\phi_1/\gamma}\right) - \phi_1 \tag{3}$$

$$\begin{aligned} f_2 = {} & \phi_1 - (1+\beta_2)\phi_2 + (1-R)\times \\ & [D/10 - \beta_1\phi_1 - (1+\beta_2)\phi_2]\exp\left(\frac{10\phi_2}{1+10\phi_2/\gamma}\right) \end{aligned} \tag{4}$$

in two unknowns ϕ_1 and ϕ_2. The unknowns represent the dimensionless temperatures in the two reactors, and are both bounded in the unit interval, i.e. $\phi_i \in [0,1]$, $i = 1,2$.

Floudas et. al. [10] state that ''solving this system can be regarded as a challenging problem.'' The parameters γ, D, β_1, and β_2 are set to 1000, 22, 2, and 2, respectively. As the recycle ratio parameter R takes on values in the set $\mathfrak{R} = \{0.935, 0.940, \dots, 0.995\}$, the number of known solutions to this system varies between 1 and 7. Using system (3) – (4), we solved the corresponding optimization problem with modified objective function (2). For each value of the parameter R given in the set $\mathfrak{R}$, we ran the C-GRASP heuristic 100 times, each time searching for all of the global minimizers. Table 1 shows the results from these runs. The second column in the table lists the number of known roots for each value of the recycle ratio parameter R. Columns three and four display the average number of roots found by C-GRASP and the time needed for C-GRASP to find those roots, respectively. As can be seen from the table, C-GRASP almost always finds all of the roots to the original system in less than 3 seconds of running time.

The next problem we consider concerns the kinematic synthesis mechanism for automotive steering. This problem is originally described in [24], and is a test problem in [20]. The Ackerman steering mechanism [26] is a four-bar mechanism for steering four wheel vehicles. When a vehicle turns, the steered wheels need to be angled so that they are both 90° with respect to a certain line. This means that the wheels will have to be at different angles with respect to the non-steering wheels.

The Ackerman design arranges the wheels automatically by moving the steering pivot inward. Pramanik [24] states that ''the Ackerman design reveals progressive deviations

TABLE 1. C-GRASP results for different recycle ratios, R. The second column is the true number of solutions. Columns 3 and 4 are the average number of solutions found and time to find those solutions over 100 C-GRASP runs.

R	# Solutions	Avg. # Found	Avg. Time
0.935	1	1.00	0.60s
0.940	1	1.00	0.77s
0.945	3	3.00	0.19s
0.950	5	4.99	1.11s
0.955	5	5.00	1.69s
0.960	7	6.96	2.41s
0.965	5	4.95	1.81s
0.970	5	4.99	1.34s
0.975	5	4.96	1.83s
0.980	5	4.98	1.90s
0.985	5	4.99	2.23s
0.990	1	1.00	0.01s
0.995	1	1.00	0.01s

from ideal steering with increasing ranges of motion.'' Pramanik instead considers a six-member mechanism. This produces the system of equations given, for $i = 1, 2, 3$, by

$$\begin{aligned}(5)\quad G_i(\psi_i,\phi_i) = &\Big[E_i(x_2\sin(\psi_i)-x_3)-F_i(x_2\sin(\phi_i)-x_3)\Big]^2+\\ &\Big[F_i(1+x_2\cos(\phi_i))-E_i(x_2\cos(\psi_i)-1)\Big]^2-\\ &\Big[(1+x_2\cos(\phi_i))(x_2\sin(\psi_i)-x_3)x_1-\\ &\qquad(x_2\sin(\phi_i)-x_3)(x_2\cos(\psi_i)-x_3)x_1\Big]^2,\end{aligned}$$

where

$$E_i = x_2(\cos(\phi_i)-\cos(\phi_0)) - x_2x_3(\sin(\phi_i)-\sin(\phi_0)) - (x_2\sin(\phi_i)-x_3)x_1$$

and

$$F_i = -x_2\cos(\psi_i) - x_2x_3\sin(\psi_i) + x_2\cos(\psi_0) + x_1x_3 + (x_3-x_1)x_2\sin(\psi_0).$$

The unknowns are x_1 (representing the normalized steering pivot rod radius), x_2 (representing the normalized tire pivot radius), and x_3 (representing the normalized 'length' direction distance from the steering rod pivot point to the tire pivot). The input parameters are the angles ψ_i and ϕ_i, for $i = 0, 1, 2, 3$.

When the angles ψ_i and ϕ_i are given as in Table 2, there are two roots to the system (5) in the domain $[0.06, 1]^3$. Using C-GRASP, we solved the corresponding optimization problem 100 times. In each run, C-GRASP was successful in finding both roots of the

TABLE 2. Angular data (in radians) for automotive steering problem.

i	ψ_i	ϕ_i
0	1.3954170041747090114	1.7461756494150842271
1	1.7444828545735749268	2.0364691127919609051
2	2.0656234369405315689	2.2390977868265978920
3	2.4600678478912500533	2.4600678409809344550

system. The average time needed to find the first root was 0.84 seconds, while 5.06 seconds were needed on average to find the second root.

The final two systems we consider are not known to be derived from real-world applications, but have been used as test systems in the literature. The first is a system of two nonlinear equations in two unknowns [20]. The system is defined by

$$f_1 = -\sin(x_1)\cos(x_2) - 2\cos(x_1)\sin(x_2) \tag{6}$$

$$f_2 = -\cos(x_1)\sin(x_2) - 2\sin(x_1)\cos(x_2). \tag{7}$$

Both variables are bounded in the interval $[0, 2\pi]$. In this domain, the system has 13 roots.

We ran C-GRASP 100 times on the corresponding optimization problem. In each run, C-GRASP was successful in finding all the roots. Figure 4 shows the time needed for the C-GRASP algorithm to find the roots to this system.

The last problem we consider is a system of two nonlinear equations taken from [10]. The system is defined by

$$\begin{aligned} f_1 &= 0.5\sin(x_1x_2) - 0.25x_2/\pi - 0.5x_1 \\ f_2 &= (1-0.25/\pi)(\exp(2x_1)-e)) + ex_2/\pi - 2ex_1, \end{aligned}$$

where $x_1 \in [0.25, 1]$ and $x_2 \in [1.5, 2\pi]$. This system has two roots in the given domain. C-GRASP was run 100 times on the corresponding optimization problem. In each run, it located both solutions, taking an average of 0.071 seconds to find the first root and 0.39 seconds to find the second.

4. CONCLUSIONS

We have considered an optimization approach to finding all roots to a system of nonlinear equations. We have given an overview of a new enhanced version of the C-GRASP heuristic for continuous global optimization, and shown the results of applying C-GRASP to four nonlinear systems from the literature. The promising results shown here, together with the previous C-GRASP results seen in [12, 14, 13] exemplify the potential of C-GRASP. While it should be noted that no one heuristic will be able to solve all problems, the results so far indicate that C-GRASP is a strong heuristic for general continuous global optimization problems.

REFERENCES

[1] E. L. Allgower and K. Georg, editors. *Computational Solution of Nonlinear Systems of Equations*. American Mathematical Society, 1990.

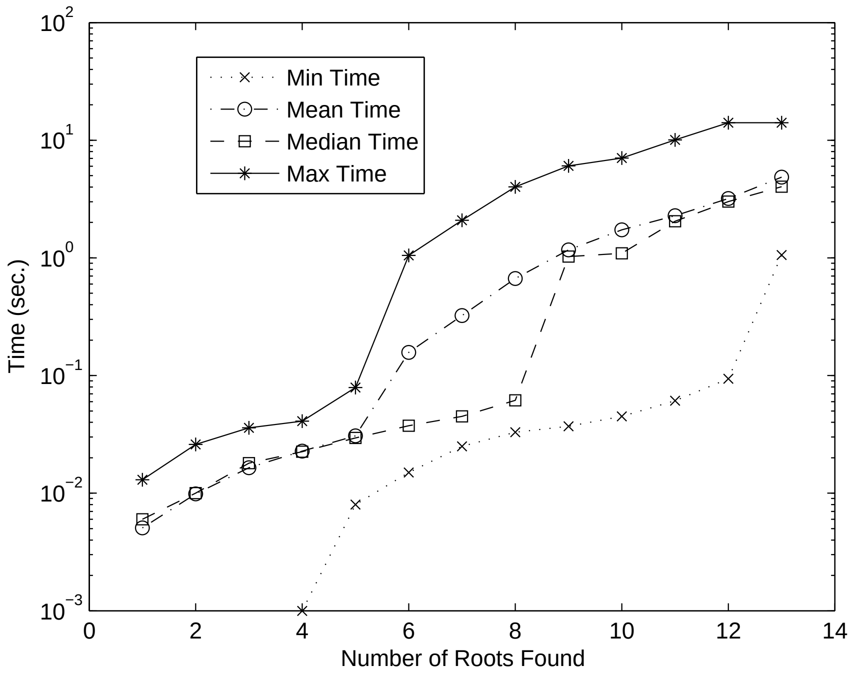

FIGURE 4. Time (in seconds) for the improved C-GRASP algorithm to find each of the roots of the system given by (6) – (7)).

[2] M. S. Bazaraa, J. J. Jarvis, and H. D. Sherali. *Linear Programming and Network Flows*. John Wiley and sons, 2nd edition, 1990.

[3] S. Blackman and N. Banh. Track association using correction bias and missing data. In O. E. Drummond, editor, *Signal and Data Processing of Small Targets (Proc. SPIE)*, volume 2235, pages 529–539, 1994.

[4] H. Cohen. *A Course in Computational Algebraic Number Theory*. Springer-Verlag, Berlin, 1993.

[5] D. A. Cox, J. B. Little, and D. O'Shea. *Ideals, Varieties, and Algorithms*. Springer-Verlag, New York, 2 edition, 1997.

[6] D. A. Cox, J. B. Little, and D. O'Shea. *Using Algebraic Geometry*. Springer-Verlag, New York, 2 edition, 2005.

[7] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6:109–133, 1995.

[8] P. Festa and M. G. C. Resende. GRASP: An annotated bibliography. In C.C. Ribeiro and P. Hansen, editors, *Essays and Surveys in Metaheuristics*, pages 325–367. Kluwer Academic Publishers, 2002.

[9] C. A. Floudas. Recent advances in global optimization for process synthesis, design, and control: enclosure of all solutions. *Computers and Chemical Engineering*, S:963–973, 1999.

[10] C. A. Floudas, P. M. Pardalos, C. Adjiman, W. Esposito, Z. Gumus, S. Harding, J. Klepeis, C. Meyer, and C. Schweiger. *Handbook of Test Problems in Local and*

Global Optimization. Kluwer Academic Publishers, Dordrecht, 1999.

[11] G. H. Golub and C. F. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[12] M. J. Hirsch, C. N. Meneses, P. M. Pardalos, and M. G. C. Resende. Global optimization by continuous GRASP. *to appear in Optimization Letters*, 2006.

[13] M. J. Hirsch, P. M. Pardalos, and M. G. C. Resende. Continuous GRASP: Enhancements and sequential stopping rules. *submitted*, 2006.

[14] M. J. Hirsch, P. M. Pardalos, and M. G. C. Resende. Sensor registration in a sensor network by Continuous GRASP. *to appear in Proc. of the IEEE Military Communications Conference (MILCOM)*, 2006.

[15] K. Hoffman and R. Kunze. *Linear Algebra*. Prentice Hall, 2nd edition, 1971.

[16] M. Kubicek, H. Hofmann, V. Hlavacek, and J. Sinkule. Multiplicity and stability in a sequence of two nonadiabatic nonisothermal CSTR. *Chemical Engineering Sceinces*, 35:987–996, 1980.

[17] M. Levedahl. An explicit pattern matching assignment algorithm. In O. E. Drummond, editor, *Signal and Data Processing of Small Targets (Proc. SPIE)*, volume 4728, pages 461–469, 2002.

[18] J. M. Martinez. Algorithms for solving nonlinear systems of equations. In E. Spedicato, editor, *Continuous Optimization: The State of the Art*, pages 81–108. Kluwer, 1994.

[19] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[20] J. P. Merlet. The COPRIN examples page. `http://www-sop.inria.fr/coprin/logiciels/ALIAS/Benches/benches.html`, 2006.

[21] J. J. Moré. A collection of nonlinear model problems. In E. L. Allgower and K. Georg, editors, *Computational Solution of Nonlinear Systems of Equations*, pages 723–762. American Mathematical Society, 1990.

[22] J. Nielson and B. Roth. On the kinematic analysis of robotic mechanisms. *The International Journal of Robotics Research*, 18(12):1147–1160, 1999.

[23] J. D. Pinter. *Computational Global Optimization in Nonlinear Systems: An Interactive Tutorial*. Lionhart Publishing, 2001.

[24] S. Pramanik. Kinematic synthesis of a six-member mechanism for automotive steering. *ASME Journal of Mechanical Design*, 124:642–645, 2002.

[25] H. L. Royden. *Real Analysis*. Macmillan Publishing, 3 edition, 1988.

[26] Wikipedia. Ackerman Steering Geometry. `http://en.wikipedia.org/wiki/Ackermann_steering_geometry`, 2006.

(Michael J. Hirsch) RAYTHEON, INC., NETWORK CENTRIC SYSTEMS, P.O. BOX 12248, ST. PETERSBURG, FL 33733-2248, AND, DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING, UNIVERSITY OF FLORIDA, 303 WEIL HALL, GAINESVILLE, FL 32611, USA.

E-mail address: `mjh8787@ufl.edu`

(Panos M. Pardalos) DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING, UNIVERSITY OF FLORIDA, 303 WEIL HALL, GAINESVILLE, FL 32611, USA.

E-mail address: `pardalos@ufl.edu`

(Mauricio G. C. Resende) ALGORITHMS AND OPTIMIZATION RESEARCH DEPARTMENT, AT&T LABS RESEARCH, 180 PARK AVENUE, ROOM C241, FLORHAM PARK, NJ 07932 USA.

E-mail address: `mgcr@research.att.com`