GRASP-BASED HEURISTICS FOR CONTINUOUS GLOBAL OPTIMIZATION PROBLEMS

By

MICHAEL J. HIRSCH

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2006

To my family

ACKNOWLEDGMENTS

The following people deserve my sincere acknowledgments: My thesis advisor, Dr. Panos Pardalos, my thesis committee (Dr. J. Cole Smith, Dr. Joseph Geunes, and Dr. William Hager), and Dr. Mauricio G. C. Resende, for their support, guidance, encouragement, and continued interest in my research; Raytheon Inc, Net-Centric Systems division, in St. Petersburg, Florida, for their financial support and commitment to my education (in particular, Mr. Stephen Marra, Mr. Gary Ridgeway, Dr. William Barker, Mrs. Linda Peak, Mr. Gifford Clegg, and Mr. Nicholas Sapankevych); Dr. Frank Grosshans, Dr. Robert Williams, and Mr. Sam Zimmerman for their guidance and support in my professional, as well as personal, life; My colleagues in the graduate school of the Industrial and Systems Engineering Department; My four grandparents, four parents, four siblings, and all my friends, for putting up with, if not always understanding, my self-imposed demanding work and research schedule.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

GRASP-BASED HEURISTICS FOR CONTINUOUS GLOBAL OPTIMIZATION PROBLEMS

By

Michael J. Hirsch

December 2006

Chair: Panagote M. Pardalos
Major Department: Industrial and Systems Engineering

In almost all areas of the applied sciences, optimization problems abound. An optimization problem can be defined as optimizing a function of several variables subject to some constraints that limit the feasible region. These problems can be defined over discrete or continuous spaces (or some combination thereof).

In global optimization, it is reasonable to assume that multiple local optima exist, different from the global optimum. Solution techniques for global optimization problems attempt to overcome locally optimal solutions in the search for a globally optimal solution. The general global optimization problem is known to be *NP-hard*. Thus, there has been significant research directed towards finding heuristics to solve global optimization problems. When very little is known about the problem structure, i.e., little or no *a priori* information, the problem can be called a *black-box* optimization problem.

This research introduces a new heuristic for continuous *black-box* global optimization problems. This heuristic is named C-GRASP, for Continuous Greedy Random Adaptive Search Procedures. In addition to fully detailing this new heuristic, we apply C-GRASP to standard global optimization test problems, as well as several challenging real-world problems.

CHAPTER 1
INTRODUCTION

Optimization problems arise in many disciplines of the natural, physical, and social sciences. Examples include materials science [105], biology, chemistry, and genetics [13, 26, 42, 66, 74, 89, 90, 91, 125, 134], military science [52, 53, 58, 70, 71, 75, 87], electrical engineering [40, 107], robotics [36, 84, 100, 102, 121], economics and portfolio theory [77], and transportation science [47, 104, 132]. Two books in particular, Floudas and Pardalos [34] and Floudas et al. [35], present an extensive list of optimization problems in the applied sciences.

In many of these problems, it is reasonable to assume that multiple local optima exist, different from the global optimum. Global optimization algorithms attempt to overcome locally optimal solutions in their search for a globally optimal solution [62, 63]. Global optimization problems can be defined over discrete or continuous spaces, and can be formulated as maximization or minimization problems. However, it is easy to see that any maximization problem can be rewritten as a minimization problem [5]. Hence, without loss of generality, we will only consider minimization problems. The global minimization problem can be stated mathematically as finding a solution $x^* \in S \subseteq \mathbb{R}^n$ such that $f(x^*) \leq f(x)$, $\forall\, x \in S$, where S is some region of $\mathbb{R}^n$ and the objective function f is defined by $f : S \rightarrow \mathbb{R}$. Such a solution x^* is called a *global minimum*. In contrast, A solution x' is a *local minimum* in a local neighborhood $S_0 \subset S$ if $f(x') \leq f(x)$, $\forall\, x \in S_0$.

There have been many powerful local search techniques developed for optimization problems. Local search algorithms iteratively improve upon the current best solution by looking in a neighborhood of the current solution for a better solution. These local search techniques employ conditions to stop when a solution is found that is the best in a local neighborhood, hence finding a *local minimum*. Unfortunately, there is no guarantee that the

local minimum found is a global minimum. Hence, there is a need for global optimization algorithms that perform more complex searches across the feasible region. One of the main difficulties in global optimization problems lies in not 'knowing' that the current minimum is a *global minimum*, and not just a *local minimum*. And while there exist algorithms for finding a *local minimum* in a finite number of iterations, it is well known that the global optimization problem is inherently unsolvable in a finite number of steps [8, 133].

Algorithms for global optimization can be classified as deterministic or random [24]. Examples of deterministic methods, which do not have any stochastic components, include branch-and-bound [33, 103], covering methods [27, 119], and generalized descent methods [119]. Random, or stochastic, methods are algorithms that employ randomness in some way to aid in finding the global optimum. The simplest such method is pure random search [8, 109, 119]. In this algorithm, a set of feasible solutions is generated randomly, according to some specified distribution (most often the uniform distribution). The objective function is evaluated at each generated solution, and the solution with smallest objective value is returned as the answer. It is easy to see that if an ε-*region* about the global minimizer has positive measure, then the probability of the pure random search algorithm choosing a solution in the ε-*region* converges to 1 [8, 25, 44, 119]. However, despite this probabilistic convergence, in practice the pure random search algorithm is very inefficient, requiring investigation of too many feasible solutions to be of practical value.

In addition to global optimization algorithms being classified as deterministic or random, different algorithms can make assumptions on the amount of available *a priori* information. On the objective function, algorithms might make the assumption that the gradient or Hessian is available, or that the objective function satisfies some regularity conditions (e.g., convexity, continuity, etc.). In addition, some algorithms might utilize information on the feasible region (e.g., polyhedral, convex, etc.). Algorithms that make assumptions on the amount of available *a priori* information on a problem will thus not be applicable to problems that do not possess this *a priori* information. Therefore, for a global optimization

algorithm, the less information needed about the structure of the problem, the more widespread its applicability. We focus on algorithms that only need information on the bounds of the variables and a mechanism for evaluating the objective function (while possibly not knowing the function analytically). These algorithms, or heuristics, can be looked at as *black-box* algorithms.

In recent years, a few classes of heuristics, originally designed for combinatorial optimization, have been extended to the domain of *black-box* continuous global optimization. The most common are simulated annealing [9, 17, 48, 76, 115, 124], genetic algorithms (including evolutionary programs) [23, 43, 49, 69, 88, 93, 116, 117, 126], and tabu search [20, 21, 50, 96]. Simulated annealing can be seen as a random walk over the feasible region, where from one iteration to another, there is a probability associated with moving to a randomly chosen new solution as opposed to staying at the current solution. This probability is a function of the objective function evaluated at the current and new solutions. In genetic algorithms, each member of a population performs a random walk over the feasible region, and at certain times in the algorithm, the population evolves, based on the current fitness of each member of the population. Tabu search also consists of a random walk over the feasible region, but a list is kept containing tabu solutions, i.e., solutions in which the algorithm is not allowed to visit. Solutions are usually kept on the tabu list for a certain time period, and then removed, allowing the algorithm to again have the ability to visit the removed solution. A good introduction to these three algorithm classes is given in Pham and Karaboga [105]. Recently, there has been the introduction of scatter search heuristics [73] (similar to genetic algorithms, but less randomization involved), as well as hybrid approaches (combining characteristics from more than one of the above classes) [120, 130]. Also, there has been an influx of papers, books, and web pages dedicated exclusively to test problems for optimization algorithms [34, 35, 92, 99, 101].

This study presents my effort in creating a new algorithm, C-GRASP, for continuous *black-box* global optimization. C-GRASP stands for Continuous Greedy Random Adaptive

Search Procedures. It can be seen as the continuous version of the combinatorial optimization heuristic GRASP [31, 32, 108]. We discuss the C-GRASP algorithm in detail, compare it to other heuristics for continuous *black-box* global optimization, and apply C-GRASP to some challenging real-world applications from across the applied science spectrum.

In the following paragraphs we describe our contributions. Each chapter of this work consists of a paper, or multiple papers, that were submitted to some refereed journal or international conference.

Chapter 2: Continuous GRASP: Algorithm and Theory. The results in this chapter appear in Hirsch et al. [55, 60]. This chapter discusses in detail two versions of the new C-GRASP algorithm for continuous global optimization. Computational experiments highlight the performance of both techniques. The better of the two C-GRASP versions is then compared with other heuristics from the literature. In addition, we also examine the performance when sequential stopping rules are implemented.

Chapter 3: Solutions to Nonlinear Systems of Equations. The contents of this chapter appear in Hirsch et al. [55, 59]. This chapter presents a technique for transforming a system of nonlinear equation into an optimization problem, with the goal of finding all solutions to the original system. We use the C-GRASP algorithm to find all solutions to systems of equations found throughout the literature.

Chapter 4: Sensor Registration in a Sensor Network. This chapter applies the C-GRASP heuristic to a sensor registration problem, encountered when multiple sensors share information across a network. It appears in Hirsch et al. [58]. The C-GRASP approach is shown to outperform two other algorithms from the sensor registration literature. Of note is that these two other algorithms have been fielded in military systems.

Chapter 5: Determination of Drugs Responsible for Adverse Reactions. This chapter looks at determining which drugs are responsible for adverse reactions. We formulate this as a nonlinear programming problem and utilize the C-GRASP algorithm to determine the responsible drugs. It appears in Hirsch et al. [54]. Our approach is shown to outperform the approach of Mammadov et al. [81, 82, 83] using three separate metrics.

Chapter 6: Object Recognition of points and lines with Unknown Correspondence. This chapter considers the problem of recognizing a two-dimensional projection of a three-dimensional object that is represented as a set of points and lines. When the correspondence between the object and image points and lines is unknown, this produces a mixed-integer nonlinear programming problem. We utilize C-GRASP along with a linear assignment algorithm to determine the correspondence between the points and lines in 3-D with those in 2-D. It appears in Hirsch et al. [57].

Test Environment:. Unless otherwise noted, all C-GRASP experiments were run 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. All implementations of the C-GRASP heuristic were written 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 [86].

CHAPTER 2
CONTINUOUS GRASP: ALGORITHM, THEORY, AND EXPERIMENTS

2.1 Introduction

Recently, there has been significant interest in developing metaheuristic algorithms for continuous global optimization problems without making use of derivative information (as derivatives might be unavailable, unreliable, or computationally challenging). Many of these metaheuristics (e.g., Simulated Annealing, Tabu Search, Genetic Algorithms) were originally applied to combinatorial optimization problems. However, within the last few decades, these algorithms have been modified for continuous optimization problems [4, 10, 20, 48, 50, 115, 120, 124, 129].

The first use of the Greedy Randomized Adaptive Search Procedure (GRASP) was for the solution of computationally difficult set covering problems [30]. Since then, the GRASP approach has been successfully applied to an overwhelming number of combinatorial optimization problems [31, 32, 108]. In this chapter, we introduce two versions of a continuous global optimization method called *Continuous-GRASP* (C-GRASP), which extends the GRASP approach of Feo and Resende [30, 31, 32, 108] from the domain of discrete combinatorial optimization to that of continuous global optimization. Our aim in this chapter is to propose a stochastic local search method that is simple to implement, can be applied to a wide range of problems, and that does not make use of derivative information, thus making it a well-suited approach for solving general continuous global optimization problems. Without loss of generality, we take the domain S as the hyperrectangle $S = \{x = (x_1, \ldots, x_n) \in \mathbb{R}^n : \ell \leq x \leq u\}$, where $\ell \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$ such that $u_i \geq l_i$, for $i = 1, \ldots, n$. The minimization problem considered is thus:

Find $x^* = \text{argmin}\{f(x) \mid \ell \leq x \leq u\}$, where $f : \mathbb{R}^n \to \mathbb{R}$, and $\ell, x, u \in \mathbb{R}^n$.

```
procedure GRASP(Problem Instance)
1     InputInstance();
2     while Stopping criteria not met do
3           ConstructGreedyRandomizedSolution(Solution);
4           LocalSearch(Solution);
5           if Solution better than BestSolution then
6                 UpdateBestSolution(Solution,BestSolution);
7           end if
8     end while
9     return(BestSolution);
end GRASP;
```

Figure 2–1: High-level pseudo-code for GRASP.

The chapter is organized as follows. Section 2.2 gives a high-level overview of the general GRASP algorithm. In Section 2.3, we provide a detailed description of the main components of the original C-GRASP algorithm [55]. The main components of the enhanced C-GRASP algorithm [60] are described in Section 2.4. In this section, we also highlight the differences between the original and enhanced C-GRASP algorithms. Section 2.5 details auxiliary procedures that the original and/or enhanced C-GRASP algorithms utilize. In Section 2.6, we present some computational experiments performed on both C-GRASP algorithms. Final remarks are given in Section 2.7.

2.2 GRASP Overview

Feo and Resende [30, 31] 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 construction. The best solution found over all of the multi-start iterations is retained as the final solution. Figures 2–1 to 2–3 provide high-level pseudo-code of the main GRASP algorithm, as well as the construction and local search phases. GRASP has been previously applied to numerous discrete combinatorial optimization problems [32]. This research describes its first application to the domain of continuous global optimization.

```
procedure ConstructGreedyRandomizedSolution(Problem Instance)
1     Solution ← ∅;
2     while Solution construction not done do
3           MakeRCL(RCL);
4           S ← SelectRandomElement(RCL);
5           Solution ← Solution ∪ {S};
6           AdaptGreedyFunction(S);
7     end while
8     return(Solution);
end ConstructGreedyRandomizedSolution;
```

Figure 2–2: High-level pseudo-code for GRASP construction procedure.

```
procedure LocalSearch(Solution, Neighborhood))
1     Solution* ← Solution;
2     while Solution* not locally optimal do
3           Solution ← SelectRandomElement(Neighborhood(Solution*));
4           if Solution better than Solution* then
5                 Solution* ← Solution;
6           end if
7     end while
8     return(Solution*);
end LocalSearch;
```

Figure 2–3: High-level pseudo-code for GRASP local search procedure.

2.3 Continuous GRASP: Original Version

In this section, we present the original C-GRASP metaheuristic for solving continuous global optimization problems subject to box constraints [55]. Section 2.3.1 describes the original C-GRASP algorithm. The original construction procedure is described in Section 2.3.2 and the original local improvement procedure in Section 2.3.3. We summarize the parameters needed for this C-GRASP version in Section 2.3.4.

2.3.1 C-GRASP Algorithm

Pseudo-code for the C-GRASP algorithm is shown in Figure 2–4. The procedure takes as input the problem dimension n, lower and upper bound vectors ℓ and u, the objective function $f(\cdot)$, as well as the parameters `MaxIter`, `MaxNumIterNoImprov`, `MaxDirToTry`, `NumTimesToRun`, and α.

In line 1 of the pseudo-code, the objective function value of the best solution found, f^*, is initialized to infinity. C-GRASP is a multi-start procedure. It is repeated `NumTimesToRun` times in the loop from line 2 to line 21. At each iteration, in line 3, an initial solution x

```
procedure C-GRASP(n, ℓ, u, f(·), MaxIters, MaxNumIterNoImprov, NumTimesToRun, MaxDirToTry, α)
1     f* ← ∞;
2     for j = 1, ..., NumTimesToRun do
3         x ← UnifRand(ℓ, u);
4         h ← 1;
5         NumIterNoImprov ← 0;
6         for Iter = 1, ..., MaxIters do
7             x ← ConstructGreedyRandomized(x, f(·), n, h, ℓ, u, α);
8             x ← LocalImprovement(x, f(·), n, h, ℓ, u, MaxDirToTry);
9             if f(x) < f* then
10                x* ← x;
11                f* ← f(x);
12                NumIterNoImprov ← 0;
13            else
14                NumIterNoImprov ← NumIterNoImprov + 1;
15            end if
16            if NumIterNoImprov ≥ MaxNumIterNoImprov then
17                h ← h/2;     /* make grid more dense */
18                NumIterNoImprov ← 0;
19            end if
20        end for
21    end for
22    return(x*);
end C-GRASP;
```

Figure 2–4: Pseudo-code for original C-GRASP algorithm.

is set to a random solution distributed uniformly over the box in $\mathbb{R}^n$ defined by ℓ and u. The parameter h that controls the discretization of the search space is initialized to 1 in line 4. The construction and local improvement phases are then called `MaxIters` times, in sequence, in lines 7 and 8, respectively.

The new solution found after local improvement is compared against the current best solution in line 9. If the new solution has a smaller objective value than the current best solution, then, in lines 10 to 12, the current best solution is updated with the new solution, and the variable `NumIterNoImprov`, which controls the search grid density, is reset to 0. Otherwise, in line 14, `NumIterNoImprov` is increased by one. If `NumIterNoImprov` becomes larger than `MaxNumIterNoImprov`, the grid density is increased in line 17, where h is halved and `NumIterNoImprov` is reset to 0 (in line 18). This allows C-GRASP to start with a coarse discretization and adaptively increase the density as needed, thereby

```
procedure ConstructGreedyRandomized(x, f(·), n, h, ℓ, u, α)
1     UnFixed ← {1, 2, ..., n};
2     while UnFixed ≠ ∅ do
3           min ← +∞;
4           max ← −∞;
5           for i = 1, ..., n do
6                 if i ∈ UnFixed then
7                       z_i ← LineSearch(x, h, i, n, f(·), ℓ, u);
8                       g_i ← f(x̌^i);
9                       if min > g_i then min ← g_i;
10                      if max < g_i then max ← g_i;
11                end if
12          end for
13          RCL ← ∅;
14          for i = 1, ..., n do
15                if i ∈ UnFixed and g_i ≤ (1 − α) ∗ min + α ∗ max then
16                      RCL ← RCL ∪ {i};
17                end if
18          end for
19          j ← RandomlySelectElement(RCL);
20          x_j ← z_j;
21          UnFixed ← UnFixed \ {j};
22    end while
23    return(x);
end ConstructGreedyRandomized;
```

Figure 2–5: Pseudo-code for original C-GRASP construction procedure.

intensifying the search in a more dense discretization when a good solution has been found. The best solution found, over all `NumTimesToRun` iterations, is returned.

2.3.2 C-GRASP Construction

The construction phase (see pseudo-code in Figure 2–5) takes as input a solution x. We start by allowing all coordinates of x to change (i.e., they are unfixed). In turn, in line 7 of the pseudo-code, a line search is performed in each unfixed coordinate direction i of x with the other $n-1$ coordinates of x held at their current values. The value z_i for the i-th coordinate that minimizes the objective function is saved, as well as the objective function value g_i in lines 7 and 8 of the pseudo-code. N.B.: In line 9, $\check{x}^i$ equals x with i-th coordinate set to z_i.

After the line search is performed for each unfixed coordinate, in lines 13 to 18 we form a restricted candidate list (`RCL`) that contains the unfixed coordinates i whose g_i values are less than or equal to $\alpha * \texttt{max} + (1-\alpha) * \texttt{min}$, where `max` and `min` are, respectively, the

```
procedure LocalImprovement(x, f(·), n, h, ℓ, u, MaxDirToTry)
1     Improved ← true;
2     D ← ∅;
3     x* ← x;
4     f* ← f(x);
5     NumDirToTry ← min{3^n − 1, MaxDirToTry};
6     while Improved do
7           Improved ← false;
8           while |D| ≤ NumDirToTry and not Improved do
9                 Generate r ← ⌈UnifRand(1, 3^n − 1)⌉ ∉ D;
10                D ← D ∪ {r};
11                d ← Ternary′(r);
12                x ← x* + h ∗ d;
13                if ℓ ≤ x ≤ u then
14                      if f(x) < f* then
15                            x* ← x;
16                            f* ← f(x);
17                            D ← ∅;
18                            Improved ← true;
19                      end if
20                end if
21          end while
22    end while
23    return(x*);
end LocalImprovement;
```

Figure 2–6: Pseudo-code for original C-GRASP local improvement procedure.

maximum and minimum g_i values over all unfixed coordinates of x, and $\alpha \in [0,1]$ is a user defined parameter. From the RCL, in lines 19 to 21, we choose a coordinate at random, say $j \in$ RCL, set x_j to equal z_j, and then fix coordinate j of x. Choosing a coordinate in this way ensures randomness in the construction phase. We continue the above procedure until all of the n coordinates of x have been fixed. At that stage, x is returned from the construction phase.

2.3.3 C-GRASP Local Improvement

The local improvement phase (with pseudo-code shown in Figure 2–6) can be seen as *approximating* the role of the gradient of the objective function $f(\cdot)$. We make no use of gradients in C-GRASP since the gradient might not be available or efficiently computable for all problems. From a given input point $x \in \mathbb{R}^n$, the local improvement algorithm generates a set of directions and determines in which direction, if any, the objective function value improves.

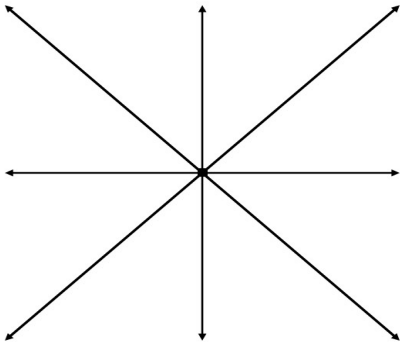

Figure 2–7: Eight directions considered in original local improvement procedure (for a two-dimensional problem).

As seen in Figure 2–7 for a problem in two dimensions (and easily generalized to n dimensions), given a point x, the eight possible directions analyzed in the local improvement algorithm are $\{(1,0),(0,1),(-1,0),(0,-1),(1,1),(-1,1),(1,-1),(-1,-1)\}$.

In general $\mathbb{R}^n$ space, each direction will be a vector of length n, with each component of this vector being one of $\{-1,0,1\}$. It is easy to see that there will thus be 3^n-1 possible directions (we exclude the case where all elements of the direction vector are 0). There is a simple mapping between each number in the set $\{1,\ldots,3^n-1\}$ and each search direction.

Let the function `Ternary` map each $r \in \mathbb{N}$ to its ternary (base 3) representation. For example, $\texttt{Ternary}(15) = 120$. If we look at a modified `Ternary` function, say $\texttt{Ternary}'$, where each $r \in \mathbb{N}$ gets mapped to its ternary representation, with each '2' being replaced by a '-1' (e.g., $\texttt{Ternary}'(15) = \{1,-1,0\}$), then the function $\texttt{Ternary}'$ maps each $r \in \mathbb{N}$ into one of the direction vectors. Therefore, rather than having to generate all direction vectors at once, it is just required to call the function $\texttt{Ternary}'$ with input a value $r \in \{1,\ldots,3^n-1\}$. The output will be the corresponding direction vector. Table 2–1 gives an example of the $\texttt{Ternary}'$ mapping when $n = 2$.

The local improvement function is given a starting solution $x \in S \subseteq \mathbb{R}^n$. The current best local improvement solution x^* is initialized to x in line 3. As seen above, for even

Table 2–1: Ternary′ mapping ($n = 2$).

i	Ternary(i)	Ternary′(i)
1	01	$\{0,1\}$
2	02	$\{0,-1\}$
3	10	$\{1,0\}$
4	11	$\{1,1\}$
5	12	$\{1,-1\}$
6	20	$\{-1,0\}$
7	21	$\{-1,1\}$
8	22	$\{-1,-1\}$

moderate values of n, the number of possible search directions can be quite large. To keep the local improvement procedure tractable, we set the variable NumDirToTry equal to the minimum of $3^n - 1$ and a user-defined parameter MaxDirToTry in line 5. Starting at the solution x^*, in the loop from lines 8 to 21, we construct up to NumDirToTry distinct random directions, in turn. In line 11, direction d is constructed and the test solution $x = x^* + h * d$ is formed, where h is the parameter that controls the density of the discretization. If the test solution x is feasible and is better than x^*, then x^* is set to x and the process restarts with x^* as the starting solution. It is important to note that the set of directions chosen can change each time through this process, as well as the order in which these directions are considered. Local improvement is terminated upon finding a solution x^* with $f(x^*) \leq f(x^* + h * d)$ for each of the NumDirToTry directions d chosen.

2.3.4 Summary of Parameters

We recap the parameters needed for the original C-GRASP heuristic.

- MaxIter defines the number of iterations per multi-start;
- MaxNumIterNoImprov is the number of iterations to perform without improvement in the best solution before increasing the discretization density;
- MaxDirToTry defines an upper bound on the number of directions to try in the local improvement procedure;
- NumTimesToRun defines the number of multi-starts to perform;
- α determines the size of the RCL;

2.4 Continuous GRASP: Enhanced Version

As seen in Section 2.3, C-GRASP works by discretizing the domain into a uniform grid. Both the construction and local improvement procedures explore solutions on this grid. As the algorithm progresses, the grid adaptively becomes more dense.

In this section, we present enhancements to the C-GRASP algorithm for solving continuous global optimization problems [60]. Section 2.4.1 details the enhanced C-GRASP algorithm. Section 2.4.2 describes the enhanced construction procedure. And in Section 2.4.3 we describe the enhanced local improvement procedure. We then summarize the parameters needed for this enhanced version in Section 2.4.4. Note that at the end of each of Sections 2.4.1-2.4.3, we discuss the appropriate differences between the original and enhanced algorithm.

2.4.1 C-GRASP Algorithm

Pseudo-code for C-GRASP is shown in Figure 2–8. The procedure takes as input the problem dimension n, lower and upper bound vectors ℓ and u, the objective function $f(\cdot)$, as well as the parameters h_s, h_e, and ρ_{lo}. h_s and h_e define the starting and ending grid discretization densities. The parameter ρ_{lo} denotes the probability that the solution returned from the local improvement procedure is an *h-local minimum* (see Section 2.4.3).

Line 1 of the pseudo-code initializes the objective function value of the best solution found to infinity. Since C-GRASP is a multi-start procedure, it is continued indefinitely, until some stopping criteria are satisfied. These stopping criteria could be based on the total number of function evaluations performed, the time elapsed since the start of the algorithm, etc. Since different implementations of C-GRASP will have different stopping criteria, we list line 2 in general form.

Each time the stopping criteria of line 2 are not satisfied, another multi-start iteration takes place, as seen in lines 2 to 18. At the beginning of each multi-start iteration, in line 3, the initial solution x is set to a random solution distributed uniformly over the box in $\mathbb{R}^n$ defined by ℓ and u. The parameter h, that controls the discretization density of the search

```
procedure C-GRASP(n, ℓ, u, f(·), h_s, h_e, ρ_lo)
1     f* ← ∞;
2     while Stopping criteria not met do
3           x ← UnifRand(ℓ, u);
4           h ← h_s;
5           while h ≥ h_e do
6                 Impr_C ← false;
7                 Impr_L ← false;
8                 [x, Impr_C] ← ConstructGreedyRandomized(x, f(·), n, h, ℓ, u, Impr_C);
9                 [x, Impr_L] ← LocalImprovement(x, f(·), n, h, ℓ, u, ρ_lo, Impr_L);
10                if f(x) < f* then
11                      x* ← x;
12                      f* ← f(x);
13                end if
14                if Impr_C = false and Impr_L = false then
15                      h ← h/2;     /* make grid more dense */
16                end if
17          end while
18    end while
19    return(x*);
end C-GRASP;
```

Figure 2–8: Pseudo-code for enhanced C-GRASP algorithm.

space, is re-initialized to h_s. The construction and local improvement phases are then called iteratively in lines 8 and 9, respectively. The solution returned from the local improvement procedure is compared against the current best solution in line 10. If the returned solution has a smaller objective value than the current best solution, then, in lines 11 to 12, the current best solution is updated with the returned solution. In line 14, if the variables $\texttt{Impr}_\texttt{C}$ and $\texttt{Impr}_\texttt{L}$ are still set to `false` then the grid density is increased by halving h, in line 15. N.B.: As seen in Section 2.4.2, the variable $\texttt{Impr}_\texttt{C}$ is `false` upon return from the construction procedure if and only if the solution input to construction equals the solution output from construction. Also, Section 2.4.3 shows that the $\texttt{Impr}_\texttt{L}$ variable is `false` on return from the local improvement procedure if and only if the input solution to local Improvement is determined to be an *h-local minimum* with probability ρ_{lo}. We increase the grid density at this stage because, as seen in Section 2.4.2, repeating the construction procedure with the same grid density will not improve the solution.

As stated in Section 2.3.1 for the original C-GRASP algorithm, and still true here, this approach allows C-GRASP to start with a coarse discretization and adaptively increase the

density as needed, thereby intensifying the search in a more dense discretization when a good solution has been found. The best solution found, at the time the stopping criteria are satisfied, is returned.

The main differences between this enhanced version of the C-GRASP algorithm and the original version presented in Section 2.3.1 deal with the input parameters and when the grid density is increased. In Section 2.3.1, the input parameters were `MaxIters`, the number of iterative cycles in each of the multi-start runs, and `MaxNumIterNoImprov`, the number of iterations without improvement until increasing the grid density. These parameters allowed the code to run for more iterations than might be otherwise necessary. The parameters h_s and h_e for the current enhanced algorithm have a more intuitive representation for the accuracy of the solution. Also, increasing the grid density when the variables $\texttt{Impr}_\texttt{C}$ and $\texttt{Impr}_\texttt{L}$ are still set to `false` in line 14 saves the code from performing unnecessary iterations that have no chance for improvement at the current discretization level.

2.4.2 C-GRASP Construction

In this section, we describe in detail the enhanced C-GRASP construction procedure. As stated above, the construction algorithm combines greediness and randomization to produce a diverse set of good-quality solutions from which to start the local improvement phase.

The construction procedure is shown in Figure 2–9. The input is a solution vector x. To start, the algorithm allows all coordinates of x to change (i.e., they are unfixed). If the `ReUse` parameter is `false` (in line 9), then in turn, in line 10 of the pseudo-code, a line search is performed in each unfixed coordinate direction i of x with the other $n-1$ coordinates of x held at their current values. The value z_i for the i-th coordinate that minimizes the objective function is saved, as well as the objective function value g_i, in lines 10 and 11 of the pseudo-code. N.B.: In line 11, $\check{x}^i$ is equal to x, with the i-th coordinate set to z_i.

```
procedure ConstructGreedyRandomized(x, f(·), n, h, ℓ, u, Impr_C)
1     UnFixed ← {1, 2, ..., n};
2     α ← UnifRand(0.0, 1.0);
3     ReUse ← false;
4     while UnFixed ≠ ∅ do
5           min ← +∞;
6           max ← −∞;
7           for i = 1, ..., n do
8                 if i ∈ UnFixed then
9                       if ReUse = false then
10                            z_i ← LineSearch(x, h, i, n, f(·), ℓ, u);
11                            g_i ← f(x^i);
12                      end if
13                      if min > g_i then min ← g_i;
14                      if max < g_i then max ← g_i;
15                end if
16          end for
17          RCL ← ∅;
18          Threshold ← min + α ∗ (max − min);
19          for i = 1, ..., n do
20                if i ∈ UnFixed and g_i ≤ Threshold then
21                      RCL ← RCL ∪ {i};
22                end if
23          end for
24          j ← RandomlySelectElement(RCL);
25          if x_j = z_j then
26                ReUse ← true;
27          else
28                x_j ← z_j;
29                ReUse ← false;
30                Impr_C ← true;
31          end if
32          UnFixed ← UnFixed \ {j};     /* Fix coordinate j. */
33    end while
34    return(x,Impr_C);
end ConstructGreedyRandomized;
```

Figure 2–9: Pseudo-code for enhanced C-GRASP construction procedure.

After looping through all unfixed coordinates (lines 7 to 16), in lines 17 to 23 we form a restricted candidate list (RCL) that contains the unfixed coordinates i whose g_i values are less than or equal to $\texttt{min} + \alpha * (\texttt{max} - \texttt{min})$, where max and min are, respectively, the maximum and minimum g_i values over all unfixed coordinates of x, and $\alpha \in [0,1]$ is randomly determined on line 2. From the RCL, in line 24, we choose a coordinate at random, say $j \in$ RCL. Line 25 checks whether x_j is equal to z_j. If this is the case, line 26 sets ReUse to the value true . Otherwise, in lines 28 to 30, ReUse is set to false , $\texttt{Impr}_\texttt{C}$ is set to true , and x_j is set to equal z_j. Finally, in line 32, we fix coordinate j of x, by removing j from the set UnFixed. Again, choosing a coordinate by selecting randomly from the current RCL ensures randomness in the construction phase. We continue the above procedure until all of the n coordinates of x have been fixed. At that stage, x and $\texttt{Impr}_\texttt{C}$ are returned from the construction phase.

At this point, it is worthwhile to examine two aspects of the enhanced construction procedure. The first concerns the ReUse parameter. Suppose at some stage in the construction procedure, we get to line 25 and determine that $x_j = z_j$. Then, when we had gone through lines 7 to 16 just before this stage, for each $i \in \texttt{UnFixed} \setminus \{j\}$, we performed a line search along the i-th coordinate, keeping all other coordinates of x fixed at their present values. So, for each $i \in \texttt{UnFixed} \setminus \{j\}$, the value of x_i is changing during the line search, but the value of x_j stays the same. The fact that line 25 is true implies that the next time through lines 7 to 16, with j no longer in UnFixed, the j-th coordinate of x has not changed from the previous time through lines 7 to 16. Hence, performing a line search in the remaining unfixed coordinates of x will produce the same results as the previous time. Therefore, setting ReUse to true allows the construction procedure to reuse the z_i and g_i values computed previously. This has the effect of speeding up the code (less calls to the line search procedure imply less function evaluations). We summarize this result in Proposition 2.1.

Proposition 2.1. *If, at any stage of the enhanced construction procedure (Figure 2–9), line 25 is true (i.e., $z_j = x_j$), then the values of z_i and g_i, $\forall\, i \in$ `UnFixed` $\setminus \{j\}$, will remain valid the next time through the for loop in lines 7 to 16.*

The second item worth examining was initially brought up in Section 2.4.1, when reference was made to halving h in the C-GRASP algorithm. Suppose $\hat{x}$ was the input to the construction procedure and $\hat{x}$ was also the output of the construction procedure. Lets examine what will happen if, using the same discretization value, $\hat{x}$ is again input to the construction procedure. For the i-th coordinate direction, keeping the other $n-1$ coordinates of $\hat{x}$ fixed, the i-th coordinate that minimizes the objective function is $\hat{x}_i$. Therefore, the first time through lines 7 to 16 of the construction procedure, z_i will be set equal to $\hat{x}_i$ and g_i will be set equal to $f(\hat{x})$ for each $i \in \{1,\ldots,n\}$. Thus, $\hat{x}$ will not be improved upon in this call to the construction procedure. This is summarized in Proposition 2.2.

Proposition 2.2. *Let the current grid discretization parameter be h. If the input to the enhanced construction procedure (Figure 2–9), $\hat{x}$, equals the output of the enhanced construction procedure, then $f(\hat{x}) \leq f(x')\ \forall\, x' \in \{x \mid \ell \leq x \leq u,\ x = \hat{x} + \gamma * h * e_i,\ \gamma \in \mathbb{Z},\ i \in \{1,\ldots,n\}\}$, where e_i is the unit vector with 1 in the i-th coordinate.*

The differences between the original construction procedure found in Section 2.3.2 and the enhanced current version are the use of the `ReUse` variable (thus saving time and function evaluations in performing unnecessary line searches) as well as allowing α to be a variable rather than a fixed input parameter. α controls the size of the `RCL`, so different α values throughout the C-GRASP run allow some construction procedures to act in a greedy fashion, while others act in a random fashion. This increases the chance of visiting different minima, hence increasing the chance of finding the global minimum.

2.4.3 C-GRASP Local Improvement

The enhanced local improvement phase (with pseudo-code shown in Figure 2–10), as with the original local improvement phase of Section 2.3.3, can be seen as *approximating*

```
procedure LocalImprovement(x, f(·), n, h, ℓ, u, ρ_lo, Impr_L)
1     x* ← x;
2     f* ← f(x);
3     NumGridPoints ← ∏_{i=1}^{n} ⌈(u_i − ℓ_i)/h⌉;
4     MaxPointsToExamine ← ⌈ρ_lo * NumGridPoints⌉;
5     NumPointsExamined ← 0;
6     while NumPointsExamined ≤ MaxPointsToExamine do
7           NumPointsExamined ← NumPointsExamined + 1;
8           x ← RandomlySelectElement(B_h(x*));
9           if ℓ ≤ x ≤ u and f(x) < f* then
10                x* ← x;
11                f* ← f(x);
12                Impr_L ← true;
13                NumPointsExamined ← 0;
14          end if
15    end while
16    return(x*, Impr_L);
end LocalImprovement;
```

Figure 2–10: Pseudo-code for enhanced C-GRASP local improvement procedure.

the role of the gradient of the objective function $f(\cdot)$. From a given input solution $x \in \mathbb{R}^n$, the local improvement procedure generates a neighborhood and determines at which solutions in the neighborhood, if any, the objective function value improves.

For the enhanced local improvement procedure, we define the concept of an *h-neighborhood* and an *h-local minimum*, where h is the current grid discretization parameter. Let $\mathtt{S}_h(\bar{x}) = \{x \in S \mid \ell \leq x \leq u,\ x = \bar{x} + \tau * h,\ \tau \in \mathbb{Z}^n\}$. It is easy to see that the points in $\mathtt{S}_h(\bar{x})$ correspond to those points in S that, in each of the coordinate directions, are integer steps (of size h) away from $\bar{x}$. Define $\mathtt{B}_h(\bar{x}) = \{x \in S \mid x = \bar{x} + \frac{h}{\|x' - \bar{x}\|}(x' - \bar{x}),\ x' \in \mathtt{S}_h(\bar{x}) \setminus \{\bar{x}\}\}$. The points of $\mathtt{B}_h(\bar{x})$ are precisely the projection of the points in $\mathtt{S}_h(\bar{x}) \setminus \{\bar{x}\}$ onto the hyper-sphere centered at $\bar{x}$ of radius h. Thus, we define the *h-neighborhood* of the solution $\bar{x}$ as the set of solutions in $\mathtt{B}_h(\bar{x})$, and $\bar{x}$ is said to be an *h-local minimum* if $f(\bar{x}) \leq f(x)$ for all $x \in \mathtt{B}_h(\bar{x})$.

The local improvement procedure is given a starting solution $x \in S \subseteq \mathbb{R}^n$. The current best local improvement solution x^* is initialized to x in line 1. Lines 3 and 4 determine the number of grid points, based on the current value of the discretization parameter h, and the maximum number of points in $\mathtt{B}_h(x^*)$ to examine to ensure x^* is an *h-local minimum* with probability ρ_{lo}, a user-supplied parameter.

Starting at the solution x^*, in the loop from lines 6 to 15, we randomly select up to `MaxPointsToExamine` solutions in $\mathtt{B}_h(x^*)$, in turn. In line 9, if the current solution selected from $\mathtt{B}_h(x^*)$, x, is feasible and is better than x^*, then x^* is set to x, $\mathtt{Impr}_\mathtt{L}$ is set to `true` , and the process restarts with the new x^* as the starting solution. N.B.: The parameter $\mathtt{Impr}_\mathtt{L}$ is used in the main C-GRASP algorithm to determine whether the local improvement procedure improved upon the solution input to the procedure. The local improvement procedure is terminated upon finding a solution x^* that is an *h-local minimum* with probability ρ_{lo}. At that time, x^* and $\mathtt{Impr}_\mathtt{L}$ are returned from the local Improvement procedure to the main C-GRASP algorithm.

The differences between the original local improvement procedure in section 2.3.3 and the enhanced version in this section concern the neighborhood of the current solution and the number of solutions in the neighborhood to examine. Suppose the current solution is $\bar{x}$. Then, the original local improvement procedure would examine up to `MaxDirToTry` solutions of the form $\bar{x}+h*\{-1,0,1\}^n$, where `MaxDirToTry` was a user-supplied parameter. Hence, only solutions on the current grid had the possibility to be examined. But there is no reason to assume that a local minimum will occur on the current grid. With the local improvement procedure described in this section, the possible solutions to be examined are the current grid points projected onto a hyper-sphere of radius h about $\bar{x}$. Therefore, points not on the current grid might also be examined. Also, the number of solutions to examine is a function of the user-defined parameter ρ_{lo} and the current number of grid points. So, in general, a larger number of solutions will be examined in the enhanced local improvement procedure.

2.4.4 Summary of Parameters

We recap the parameters needed for the enhanced C-GRASP heuristic.

- h_s defines the starting grid density;
- h_e defines the ending grid density;

```
procedure LineSearch(x, f(·), n, h, ℓ, u, k)
1     t ← x;
2     z_k ← x_k;
3     minF ← f(x);
4     t_k ← ℓ_k;
5     while t_k ≤ u_k do
6           if f(t) < minF then
7                 minF ← f(t);
8                 z_k ← t_k;
9           end if
10          t_k ← t_k + h;
11    end while
12    t_k ← u_k;     /* just in case u_k skipped. */
13    if f(t) < minF then
14          minF ← f(t);
15          z_k ← t_k;
16    end if
17    return(z_k);
end LineSearch;
```

Figure 2–11: Pseudo-code for C-GRASP line search procedure.

- ρ_{lo} defines the probability that the returned solution from the local improvement phase is an *h-local minimum*;

2.5 Continuous GRASP: Auxiliary Components

In this section we describe the auxiliary procedures that are utilized by the two versions of the C-GRASP algorithm. The line search pseudo-code is presented in Figure 2–11. The line search procedure is called from both the original and enhanced construction procedures and takes as input a solution vector x, the number of variables n, the objective function $f(\cdot)$, the current grid discretization value, h, the lower and upper bounds ℓ and u, as well as an index $k \in \{1,\ldots,n\}$. This procedure will sample the objective function as the k-th coordinate of x takes on the values $\{x_k, \ell_k, \ell_k + h, \ldots, \ell_k + \nu * h, u_k\}$, where $\nu = \max[\, \mu \mid \mu \in \mathbb{Z},\ \ell_k + \mu * h \leq u_k \,]$. This is illustrated in Figure 2–12. Returned from the line search procedure is the sampled value of the k-th coordinate that produced the minimum objective function value.

The feasible procedure is called from both the original and enhanced local improvement procedures (pseudo-code is seen in Figure 2–13). The procedure takes as input a solution

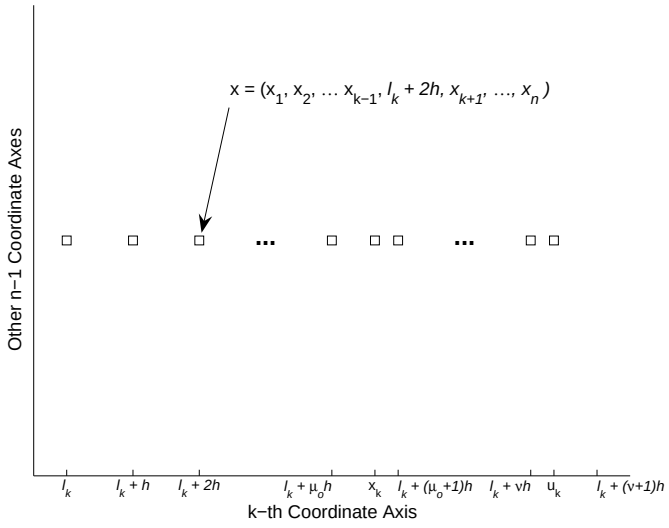

Figure 2–12: Points at which objective function is sampled during line search procedure (in k-th coordinate direction).

```
procedure Feasible(x, n, ℓ, u)
1     Feas ← true;
2     for i = 1, ..., n do
3         if x_i < ℓ_i or x_i > u_i then
4             Feas ← false;
5         end if
6     end for
7     return(Feas);
end Feasible;
```

Figure 2–13: Pseudo-code for C-GRASP feasible procedure.

vector x, the number of variables n, and the lower and upper bounds ℓ and u. Returned from the procedure is a boolean value that is set to `true` if and only if $\ell \leq x \leq u$.

The `Ternary`$'$ procedure (pseudo-code seen in Figure 2–14) is only called from the original local improvement procedure. Input is a number $r \in \mathbb{Z} \cap [1, 3^n - 1]$. Returned is the unique direction vector corresponding to r, as defined in Section 2.3.3.

2.6 Computational Results

In this section, we discuss the performance of the C-GRASP algorithms through three different experiments. First, we compare the two versions of the C-GRASP algorithm presented above. Then, we highlight the performance of the enhanced C-GRASP algorithm

```
procedure Ternary′(n, r)
1     i ← n;
2     while r ≥ 3 do
3           dir_i ←   mod [r, 3];
4           r ← ⌊r/3⌋;
5           i ← i − 1;
6     end while
7     dir_i ← r;
8     for i = 1, ..., n do
9           if dir_i == 2 then
10                dir_i ← −1;
11          end if
12    end for
13    return(dir);
end Ternary′;
```

Figure 2–14: Pseudo-code for C-GRASP Ternary′ procedure.

when compared with a genetic algorithm, a scatter search algorithm, and a tabu search algorithm. Lastly, we observe the performance of the enhanced C-GRASP algorithm when implemented with sequential stopping rules. Note that in the enhanced C-GRASP local improvement procedure, we make the restriction that MaxPointsToExamine ≤ 1000, for otherwise, large dimension test functions will take much too long to run.

2.6.1 C-GRASP Comparison

In order to justify that the enhanced C-GRASP algorithm (Section 2.4) does offer significant improvements as compared to the original C-GRASP algorithm (Section 2.3), this section compares the two C-GRASP algorithms on a set of 14 of the test functions taken from the Appendix. These functions were chosen for two reasons. First, previous papers have used these functions to test and compare their algorithms. Second, according to Hedar and Fukushima [50], ‘‘the characteristics of these test functions are diverse enough to cover many kinds of difficulties that arise in global optimization problems.’’ Since these test functions have known global minima, the two C-GRASP algorithms were run until the current objective function value, $f(x)$, was significantly close to the global optimum $f(x^*)$. The concept of significantly close is defined in Inequality 2–1, where $\varepsilon_1 = 10^{-4}$ and $\varepsilon_2 = 10^{-6}$ [48, 49, 50, 55, 60, 115].

Table 2–2: Original C-GRASP parameter values (for results in Section 2.6.1 and Table 2–4).

Parameter	Value	Parameter	Value
α	0.4	h (Starting value)	1
`MaxDirToTry`	30	`MaxIters`	200
`MaxNumIterNoImprov`	20	`NumTimesToRun`	20

Table 2–3: Enhanced C-GRASP parameter values (for results in Section 2.6.1 and Table 2–4).

Fn.	h_s	h_e	Fn.	h_s	h_e
BR	1	0.02	EA	1	0.1
GP	1	1	SH	1	0.01
$H_{3,4}$	0.5	0.05	$H_{6,4}$	0.5	0.005
R_2	1	0.01	R_5	1	0.01
R_{10}	1	0.01	$S_{4,5}$	1	0.5
$S_{4,7}$	1	0.5	$S_{4,10}$	1	0.5
Z_5	1	0.5	Z_{10}	1	0.005

$$|f(x^*) - f(x)| \leq \varepsilon_1 |f(x^*)| + \varepsilon_2, \tag{2–1}$$

For each of the 14 test functions, we ran both versions of the C-GRASP algorithm 100 times. Each run used a different starting seed for the pseudo random number generator. For both C-GRASP algorithms, the stopping criteria were finding a solution significantly close to the global optimum or completing 20 multi-starts. The parameters used for the original C-GRASP algorithm are listed in Table 2–2. For the enhanced C-GRASP algorithm, ρ_{lo} was set to 0.7 for all of the test functions. The h_s and h_e values for each test function are listed in Table 2–3.

We recorded the number of function evaluations and the elapsed time for the current solution to satisfy Inequality 2–1. The average of these results for each of the 14 test functions are shown in Table 2–4, along with the percentage of time the algorithms found a significantly close solution.

As is clear from Table 2–4, in almost all the cases, both C-GRASP versions always find the global optimum. Also, for the enhanced version, overall there has been a monumental decrease in the number of objective function evaluations needed to find a solution

Table 2–4: Original and enhanced C-GRASP comparison.

Test	Original C-GRASP			Enhanced C-GRASP		
Fn.	Fn. Evals	Time (s)	Sig. Cl. %	Fn. Evals	Time (s)	Sig. Cl. %
BR	59,857	0.0016	100	10,090	0.0011	100
EA	89,630	0.0042	100	5,093	0.0008	100
GP	29	0.0000	100	53	0.0000	100
SH	82,363	0.0078	100	18,608	0.0032	100
$H_{3,4}$	20,743	0.0026	100	1,719	0.0006	100
$H_{6,4}$	79,685	0.0140	100	29,894	0.0122	100
R_2	1,158,350	0.0132	100	23,544	0.0021	100
R_5	6,205,503	1.7520	100	182,520	0.0148	100
R_{10}	20,282,529	11.4388	99	725,281	0.2739	100
$S_{4,5}$	5,545,982	2.3316	100	9,274	0.0021	100
$S_{4,7}$	4,052,800	2.3768	100	11,766	0.0027	100
$S_{4,10}$	4,701,358	3.5172	100	17,612	0.0044	100
Z_5	959	0.0000	100	12,467	0.0025	100
Z_{10}	3,607,653	1.0346	100	2,297,937	1.1090	100

satisfying Inequality 2–1, with no impact to the percentage of successful runs. There is also a small time decrease for the enhanced C-GRASP algorithm.

2.6.2 Comparison with Other Algorithms

In this section, we compare the enhanced C-GRASP algorithm with three others from the literature. We make this comparison over 40 of the multimodal test functions listed in the Appendix, all with known global minima. The three other heuristics for continuous optimization chosen to compare against C-GRASP are: Genetic Algorithm for Numerical Optimization of COnstrained Problems (*Genocop III*) [93, 94], Scatter Search (*SS*) [73], and Directed Tabu Search (DTS_{APS}) [50].

At any time during an algorithms run, we define the optimality gap by $GAP = |f(x) - f(x^*)|$, where x is the current best solution found by the algorithm and x^* is the known global minimum solution. We then claim that the algorithm has successfully solved the problem if Inequality 2–2 is satisfied, where $\varepsilon = 0.001$ [50, 60, 73].

$$GAP \leq \begin{cases} \varepsilon & \texttt{if} \quad f(x^*) = 0 \\ \varepsilon * |f(x^*)| & \texttt{if} \quad f(x^*) \neq 0 \end{cases} \tag{2–2}$$

Table 2–5: Enhanced C-GRASP parameter values (for results in Section 2.6.2, Tables 2–6 and 2–7, and Figure 2–15).

Test Fn.	h_s	h_e	Test Fn.	h_s	h_e
BE	0.1	0.05	B_2	1	0.1
BO	0.1	0.05	BR	0.1	0.05
EA	1	0.1	GP	0.1	0.05
M	0.1	0.05	R_2	1	0.1
SC_2	5	0.25	SH	0.1	0.05
CA	0.1	0.05	Z_2	1	0.1
SP_3	0.1	0.05	$H_{3,4}$	0.1	0.05
CV	1	0.05	$P_{4,\frac{1}{2}}$	0.1	0.0125
$P^0_{4,\frac{1}{2}}$	0.1	0.05	$PS_{4,\sigma}$ [a]	0.1	0.05
$S_{4,5}$	0.1	0.05	$S_{4,7}$	0.1	0.05
$S_{4,10}$	0.1	0.05	$H_{6,4}$	0.1	0.05
SC_6	50	0.25	T_6	1	0.1
GR_{10}	10	0.25	RA_{10}	2	0.1
R_{10}	2	0.05	SS_{10}	1	0.1
T_{10}	20	0.1	Z_{10}	1	0.1
GR_{20}	10	0.25	RA_{20}	2	0.1
R_{20}	2	0.1	SS_{20}	1	0.1
Z_{20}	2	0.05	PW_{24}	2	0.1
DP_{25}	5	0.2	A_{30}	5	0.05
L_{30}	2	0.05	SP_{30}	1	0.05

[a] $\sigma = \{8, 18, 44, 114\}$.

At several points during each algorithms run, the *GAP* value was computed. For the enhanced C-GRASP algorithm, we again set ρ_{lo} equal to 0.7 for all test functions. The h_s and h_e values for each test function are listed in Table 2–5.

The average *GAP* values over all 40 test functions is listed for each algorithm in Table 2–6. N.B.: The C-GRASP results are the average *GAP* values for all test functions obtained by running the enhanced algorithm 100 times for each test function. The DTS_{APS} results were averaged over 7 runs for each test function. The results for the DTS_{APS} algorithm are found in Hedar and Fukushima [50] and the results for *Genocop III* and *SS* in Hedar and Fukushima [50] and Laguna and Martí [73]. Since the *Genocop III* algorithm performed extremely poor on function SC_6, two rows in Table 2–6 are devoted to this algorithm, one with SC_6 included in the average, and one with the results for this function not included. Also listed in this table are the enhanced C-GRASP results if the function

Table 2–6: Average *GAP* values (over all functions in Table 2–7).

f-evals.	100	500	1,000	5,000	10,000	20,000	50,000
Genocop [a]	$5.37E+25$	$2.39E+17$	$1.13E+14$	636.37	399.52	320.84	313.34
Genocop [b]	1,335.45	611.30	379.03	335.81	328.66	324.72	321.20
SS	134.45	26.34	14.66	4.96	3.60	3.52	3.46
DTS_{APS}	50,400	43.06	24.26	4.22	1.80	1.70	1.29
C-GRASP [c]	23,610.61	10,185.84	1,341.70	6.20	4.73	3.92	3.02
C-GRASP [d]	24,208.75	10,442.53	1,371.62	1.87	0.37	0.07	0.03

[a] Average values over all test problems.

[b] Average values over all test problems except problem SC_6.

[c] Average values over all test problems.

[d] Average values over all test problems except problem Z_{20}.

Z_{20} is removed. For this metric, the enhanced C-GRASP algorithm remains competitive with the other algorithms, at the 50,000 objective function evaluation mark doing better than all other algorithms except DTS_{APS}. Unfortunately, Table 2–6 does not shed light on the true performance of the algorithms. Due to having 'trouble' with one function, the average enhanced C-GRASP *GAP* values are higher than those of DTS_{APS}. Note that when function Z_{20} is removed (sixth row of Table 2–6), C-GRASP does much better than all the other algorithms from 5,000 objective function evaluations onward. To really compare these algorithms, we need to look at the performance on a function by function basis. For the C-GRASP algorithm, the average performance for each test function is presented in Table 2–7. At 50,000 objective function evaluations, all of the C-GRASP *GAP* values are less than 1 except Z_{20}, which has a *GAP* value of 119.7624. Laguna and Marti [73] state that the final *SS GAP* values are all less than 1 except for four functions: SC_6, RA_{10}, R_{20}, and A_{30}, with *GAP* values of 118.4341, 9.9496, 2.2441, and 5.5033 respectively. However, final *GAP* values for each of the individual test functions are not reported for the DTS_{APS}, *SS*, and *Genocop III* algorithms [50, 73, 93, 94].

As an alternative to comparing actual *GAP* values, Figure 2–15 displays the number of test functions solved (according to Inequality 2–2) by each algorithm as a function of

Table 2–7: Average *GAP* value of each test function for enhanced C-GRASP algorithm (bold items denote those *GAP* values satisfying Inequality 2–2).

Test Fn.	Number of Objective Function Evaluations						
	100	500	1,000	5,000	10,000	20,000	50,000
BE	0.0036	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
B_2	0.4807	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
BO	2.5982	0.0032	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
BR	0.0084	0.0035	**0.0001**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
EA	0.0089	0.0089	0.0089	**0.0004**	**0.0004**	**0.0004**	**0.0004**
GP	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
M	**0.0004**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
R_2	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
SC_2	119.4013	0.3048	0.2368	0.1612	0.0103	0.0065	**0.0001**
SH	0.2498	**0.1617**	**0.1617**	**0.1511**	**0.1511**	**0.1511**	**0.0208**
CA	0.0405	**0.0002**	**0.0002**	**0.0002**	**0.0001**	**0.0001**	**0.0001**
Z_2	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
SP_3	0.5376	**0.0010**	**0.0010**	**0.0001**	**0.0001**	**0.0001**	**0.0001**
$H_{3,4}$	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
CV	11.3623	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
$P_{4,\frac{1}{2}}$	7.2999	0.8879	0.1328	0.0359	0.0118	0.0031	0.0031
$P^0_{4,\frac{1}{2}}$	2463.2448	0.6782	0.0109	0.0012	**0.0004**	**0.0003**	**0.0000**
$PS_{4,\sigma}$ [a]	0.8433	0.0069	0.0033	**0.0007**	**0.0005**	**0.0002**	**0.0000**
$S_{4,5}$	9.2904	6.9209	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
$S_{4,7}$	9.5018	5.9017	**0.0001**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
$S_{4,10}$	9.5240	7.2325	0.8944	**0.0001**	**0.0001**	**0.0001**	**0.0001**
$H_{6,4}$	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
SC_6	835.8956	221.9143	144.6920	1.6621	1.2104	0.1214	0.0233
T_6	335.0119	192.9117	15.3233	**0.0000**	**0.0000**	**0.0000**	**0.0000**
GR_{10}	20.3141	20.3141	12.4181	**0.0000**	**0.0000**	**0.0000**	**0.0000**
RA_{10}	77.2000	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
R_{10}	8064.0275	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
SS_{10}	259.9448	36.7055	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
T_{10}	6402.6928	410.7253	64.8220	18.6801	6.6901	1.5025	0.2181
Z_{10}	37.1243	37.1243	27.9358	0.2051	0.0646	0.0059	0.0011
GR_{20}	61.0211	61.0211	61.0211	31.7019	4.4302	**0.0000**	**0.0000**
RA_{20}	258.8922	188.1984	102.4020	**0.0000**	**0.0000**	**0.0000**	**0.0000**
R_{20}	210591.5859	89313.8553	15140.9928	**0.0000**	**0.0000**	**0.0000**	**0.0000**
SS_{20}	1934.5603	1211.4429	732.6502	**0.0000**	**0.0000**	**0.0000**	**0.0000**
Z_{20}	283.0573	174.8449	174.8449	174.8449	174.8449	153.9820	119.7624
PW_{24}	5296.1979	1056.8398	434.9449	1.4250	0.1948	0.1528	0.0806
DP_{25}	707004.8655	314195.4226	36508.5904	0.6680	0.6680	0.6680	0.6680
A_{30}	19.3023	18.6985	17.5581	0.6078	**0.0000**	**0.0000**	**0.0000**
L_{30}	126.3283	111.4947	96.5772	4.1265	0.8559	0.0088	**0.0000**
SP_{30}	181.8665	159.9949	131.7025	13.5988	**0.0000**	**0.0000**	**0.0000**

[a] $\sigma = \{8, 18, 44, 114\}$.

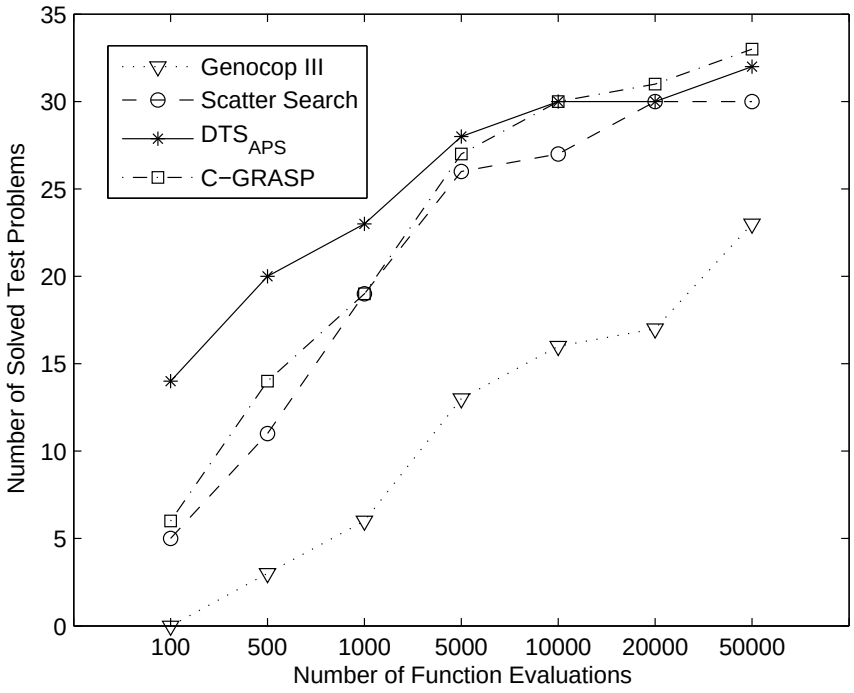

Figure 2–15: Number of solved test problems (for each algorithm) as a function of the number of objective function evaluations.

the number of objective function evaluations. This figure shows C-GRASP performing competitively with the other algorithms, solving at least as many test functions (as the other algorithms) from $10,000$ objective function evaluations onward, and solving 33 test problems when reaching $50,000$ objective function evaluations, which is one more that the DTS_{APS} algorithm.

2.6.3 Enhanced C-GRASP with Sequential Stopping Rules

In most global optimization problems, the solution is not known ahead of time. While the *significantly close* and *GAP* inequalities (Inequalities 2–1 and 2–2) are useful for comparing heuristics on problems with known global optimum, they are not useful in practice. Therefore, as a third test, we implemented the enhanced C-GRASP algorithm with the sequential stopping rules derived by Hart [44].

Hart [44] defines the general multistart random search (*MSRS*) algorithm as follows (where $L : S \rightarrow S$ is a local search algorithm that takes a sample from S and finds a sample that is a local minimum of f).

Algorithm MSRS: Let $\xi_1, \xi_2, \ldots$ be independent and identically distributed random vectors with common distribution G on S. Let $(X_1, Y_1), (X_2, Y_2), \ldots$ be defined by

Step 1. $X_1 = L(\xi_1)$ and $Y_1 = f(X_1)$

Step $k+1$. If $f(L(\xi_{k+1})) \leq Y_k$, then $X_{k+1} = L(\xi_{k+1})$ and $Y_{k+1} = f(X_{k+1})$. Otherwise,

$X_{k+1} = X_k$ and $Y_{k+1} = Y_k$.

Also, Hart defines $\hat{\rho}_r(\varepsilon) = \rho_r(\varepsilon) + \Gamma_r(\varepsilon)$, where r is the number of steps taken in the *MSRS* algorithm, $\rho_r(\varepsilon) = \sup\{k \mid \tau_k(r) > 0, Y_{\tau_k(r)} \leq Y_r + \varepsilon\}$, $\Gamma_r(\varepsilon) = |\{Y_i \mid Y_i \leq Y_r + \varepsilon,\ i = \tau_2(r)+1, \ldots, r-1\}|$, $\tau_1(r) = r$, and for $j = 2, \ldots, r$, $\tau_j(r)$ is defined by Equation 2–3.

$$\tau_j(r) = \begin{cases} 0 & \text{if } \{k \mid 1 \leq k < \tau_{j-1}(r),\ Y_k \neq Y_{\tau_{j-1}(r)}\} = \emptyset \\ \sup\{k \mid 1 \leq k < \tau_{j-1}(r), & \\ \qquad Y_k \neq Y_{\tau_{j-1}(r)}\} & \text{o.w.} \end{cases} \tag{2–3}$$

The term $\hat{\rho}_r(\varepsilon)/r$ is used to estimate $\rho_\varepsilon = \mathbf{P}(f(x) \leq f(x^*) + \varepsilon)$, the probability that $f(x)$ is within ε of the global minimum $f(x^*)$. Hart shows that $\hat{\rho}_r(\varepsilon)/r$ is a better estimate to ρ_ε than the estimate $\rho_r(\varepsilon)/r$ defined by Dorea [25]. Without too much difficulty, it can be seen from the above definitions that $\Gamma_r(\varepsilon) = r - \tau_2(r) - 1$, i.e., $\Gamma_r(\varepsilon)$ counts the number of steps whose Y value is equivalent to the current Y value, Y_r.

Hart's stopping rule is then defined as follows: For a given $\varepsilon > 0$, $\delta > 0$, and $\beta \in (0,1)$, terminate *Algorithm MSRS* if $r \geq 2$ and Inequality 2–4 is satisfied, where Φ is the cumulative distribution function of the standard normal, i.e., $\Phi(x)$ is defined as in Equation 2–5.

$$\Phi(2\delta\sqrt{r}) - \Phi(-2\delta\sqrt{r}) - (1 - \hat{\rho}_r(\varepsilon)/r)^r \geq 1 - \beta \tag{2–4}$$

Table 2–8: Average multi-start and *GAP* value for each test function when enhanced C-GRASP algorithm implemented with Hart [44] stopping rules (bold items denote those *GAP* values satisfying Inequality 2–2).

Fn.	# MS	*GAP*	Fn.	# MS	*GAP*
BE	8.4	**0.0000**	B_2	8	**0.0000**
BO	8	**0.0000**	BR	8	**0.0000**
EA	8.4	**0.0000**	GP	8.2	**0.0000**
M	8	**0.0000**	R_2	8	**0.0000**
SC_2	8.4	**0.0000**	SH	8	**0.0003**
CA	8.3	**0.0000**	Z_2	8	**0.0000**
SP_3	8.2	**0.0000**	$H_{3,4}$	8.4	**0.0000**
CV	8.4	**0.0000**	$P_{4,\frac{1}{2}}$	9.3	0.0023
$P^0_{4,\frac{1}{2}}$	8.3	**0.0000**	$PS_{4,\sigma}$ [a]	8.4	**0.0000**
$S_{4,5}$	8.5	**0.0000**	$S_{4,7}$	8.3	**0.0000**
$S_{4,10}$	8.8	**0.0001**	$H_{6,4}$	8.5	**0.0000**
SC_6	8.1	**0.0000**	T_6	8	**0.0000**
GR_{10}	9.5	**0.0000**	RA_{10}	8	**0.0000**
R_{10}	8.6	**0.0000**	SS_{10}	8	**0.0000**
T_{10}	8	0.0029	Z_{10}	8.2	**0.0000**
GR_{20}	10.1	**0.0000**	RA_{20}	8	**0.0000**
R_{20}	8	**0.0000**	SS_{20}	8	**0.0000**
Z_{20}	9	0.0049	PW_{24}	8.3	**0.0010**
DP_{25}	8.2	**0.0000**	A_{30}	8	**0.0000**
L_{30}	8	**0.0000**	SP_{30}	8	**0.0000**

[a] $\sigma = \{8, 18, 44, 114\}$.

$$\Phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{\frac{-y^2}{2}} dy \tag{2–5}$$

In this stopping rule, ε is the desired accuracy of the best solution found, δ is a limit on the difference between $\hat{\rho}_r(\varepsilon)/r$ and ρ_ε (ensures that r is sufficiently large for ρ_ε to be reliably estimated by $\hat{\rho}_r(\varepsilon)/r$), and $1-\beta$ is the required probability of success.

We implemented Hart's stopping rules with the enhanced C-GRASP algorithm and ran each of the 40 test functions 10 times. The stopping rules parameters were defined to be $\varepsilon = 0.001$, $\delta = 0.4$, and $\beta = 0.025$, and each function used the same h_s and h_e values as listed in Table 2–5. Note that with ε, δ, and β defined this way, the minimum $r \in \mathbb{Z}$ that satisfies Inequality 2–4 is $r = 8$. For each test function, we computed the average *GAP* value when the algorithm was finished. Table 2–8 presents these results for each function. Each function has associated with it two numbers: the first is the average number

of multi-starts performed over the runs, and the second is the average final *GAP* value over the runs. From this table, one can see that when the enhanced C-GRASP algorithm is used as intended, as a true multi-start algorithm, almost all of the functions have a final *GAP* value satisfying Inequality 2–2. Only T_{10}, $P_{4,\frac{1}{2}}$, and Z_{20} do not satisfy the inequality. They have *GAP* values of 0.0029, 0.0023, and 0.0049 respectively. Even these *GAP* values are quite small. Hence, all in all, the C-GRASP algorithm did extremely well against these test functions.

2.7 Conclusions

In this chapter, we have presented two version of a new GRASP heuristic for continuous global optimization. As seen in Sections 2.3 and 2.4, the algorithms are easy to describe and implement. We have shown that the enhanced C-GRASP algorithm has significant advantages over the original C-GRASP algorithm. Furthermore, the enhancements do not detract from C-GRASP being generally applicable to global optimization problems. The enhanced C-GRASP algorithm was compared against three others from the recent literature, as well as being implemented with sequential stopping rules. The encouraging results shown in this chapter demonstrate the power of the C-GRASP algorithm. Since the C-GRASP algorithm makes no use of derivative nor *a priori* information, it is a well-suited approach for solving general, continuous, *black-box* global optimization problems.

CHAPTER 3
SOLUTIONS TO NONLINEAR SYSTEMS OF EQUATIONS

3.1 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 Floudas et al. [35] and Moré [97].

If the system is linear, then there are many approaches available to determine the solutions (e.g., gaussian elimination and matrix inverses [39], null-space computation [39, 61], linear programming [5], etc.). If the system happens to be polynomial in nature, then exact techniques employing resultants [16] or Gröbner bases [18, 19] 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 in the intermediate stages, thus making them intractable for even moderate-sized problems.

When the equations in the system do not exhibit nice linear or polynomial properties, then it can be very difficult to determine any or all solutions of the system. In this general case, we are left with numerical routines to determine solutions of the system. Unfortunately, there is no one numerical method that will guarantee finding all the solutions to a general system of equations in finite time. Most numerical solution techniques are based on Newton's method [85]. Each iteration of Newton's method consists of computing the Jacobian matrix at a feasible solution and solving a linear system which approximates the original system at that solution. 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 [85], at each iteration a convex quadratic function is solved to determine the next feasible solution ay which to step. 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) [85, 102]. This approach begins with a 'starting' system of equations (not the true system) whose solution is known. The starting system is gradually transformed into the true system. At each stage in this transformation, the current system is solved to find a starting solution for the next stage system. The idea is that as the system is changing, the solution is following, or tracing, an arc from a solution of the starting system to a solution of the true original system. Each system can then be solved by a Newton-type method, again making use of the Jacobian. Problems with this approach are that not all paths will converge to a finite solution.

In 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 chapter, we transform the system of nonlinear equations into a global optimization problem. We then discuss how the global optimization problem can be modified to find all solutions of the original system. Using the enhanced C-GRASP algorithm, we demonstrate this approach on systems of equations from the literature.

3.2 Nonlinear System Transformation to Global Optimization Problem

In this section, we transform the problem of *determining all solutions to a system of equations* into a problem of *finding all global minimizers of a nonlinear 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 $i = 1, \ldots, r$. 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 $F(x)$, as

defined in Equation 3–1. We make use of Proposition 3.1 in what follows.

$$F(x) = \sum_{i=1}^{r} f_i^2(x) \tag{3–1}$$

Proposition 3.1. *Let $f_1(x), \ldots, f_r(x)$ define a system of r equations in $\mathbb{R}^n$. Let $F(x) = \sum_{i=1}^{r} f_i^2(x)\ \forall\ x\ \in \mathbb{R}^n$. (i) $x^* \in \mathbb{R}^n$ is a root of the system $f_1(x), \ldots, f_r(x) \Longrightarrow x^*$ is a global minimizer of $F(x)$. (ii) $x^* \in \mathbb{R}^n$ is a global minimizer of $F(x)$ and $F(x^*) = 0 \Longrightarrow x^*$ is a root of the system $f_1(x), \ldots, f_r(x)$. (iii) $x^* \in \mathbb{R}^n$ is a global minimizer of $F(x)$ and $F(x^*) > 0 \Longrightarrow$ the system $f_1(x), \ldots, f_r(x)$ is inconsistent.*

Therefore, the original problem of solving a system of equations can be transformed to the problem of minimizing a nonlinear function, as seen in Problem 3–2. If the global minimizer of Problem 3–2 has objective function value 0, then it is a solution to the system of equations. Otherwise, the system or equations is inconsistent.

$$\text{Find } x^* = \text{argmin}\{F(x) = \textstyle\sum_{i=1}^{r} f_i^2(x) \mid x \in \mathbb{R}^n\}. \tag{3–2}$$

We note that our goal is not to find one solution to the original system of equations, but to find all solutions. Suppose the original system has κ solutions. Solving Problem 3–2 κ times using C-GRASP (or any heuristic, for that matter) with different starting solutions gives no guarantee of finding all κ roots. It is entirely possible that some of the roots would be found multiple times, while others would not be found at all. To overcome this problem, we choose to adaptively modify the objective function $F(x)$ as roots are found. So, suppose that C-GRASP has just found the k-th root (roots are denoted $x^1, \ldots, x^k$). Then the C-GRASP algorithm will restart, with modified objective function given by Equation 3–3, where the characteristic function $\chi_\rho(\delta)$ [111] is defined in Equation 3–4, β is a large constant, and ρ is a small constant.

$$F(x) = \sum_{i=1}^{r} f_i^2(x) + \beta \sum_{j=1}^{k} e^{-\|x-x^j\|} \chi_\rho(\|x-x^j\|), \tag{3–3}$$

$$\chi_\rho(\delta) = \begin{cases} 1 & \text{if } \delta \leq \rho \\ 0 & o.w. \end{cases} \tag{3–4}$$

The 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 extremely important to choose ρ small enough so that other solutions that have yet to be found are not adversely penalized in the modified objective function.

3.3 Solving Systems of Equations with C-GRASP

In this section, we use C-GRASP to highlight the transformation and adaptive modifications in the previous section to find all solutions to systems of equations. For each real-world application considered, we provide an overview of the problem. To the authors knowledge, no derivative-free heuristic has been applied to these problems.

3.3.1 Robot Kinematics

This problem is originally briefly mentioned in Freudenstein [36]. Tsai and Morgan [121] derive a system of equations to describe the robot arm kinematics problem. In simple terms, we are given an arm of a robot that is composed of 6 rigid links. Each link is connected to no more than two others by a revolute joint. A revolute joint is a joint in which only rotational motion about the joint axis can occur. The first link is designated the base and connected to the second link, the second link is connected to the first and third links, etc. At the end, link 6 is connected to link 5 and the hand (which can also be seen as the seventh link) of the robot. A simple diagram of this representation is seen in Figure 3–1.

Associated with each pair of links i and $i+1$, we have four parameters: a_i is the length of link $i+1$, d_i is the length of joint i, α_i is the angle needed to rotate joint $i+1$ so that joint i and joint $i+1$ would be parallel, and θ_i is the angle needed to rotate link $i+1$ so that link i and link $i+1$ would be parallel. These parameters are diagramed in Figure 3–2.

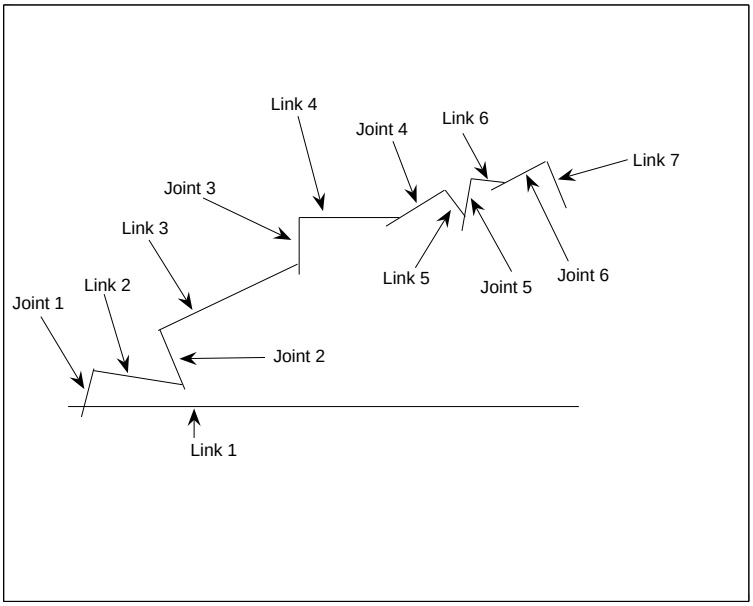

Figure 3–1: 6-revolute robot arm diagram.

Note that in Manocha and Canny [84], θ_i is called the *i-th joint rotation angle*, α_i is called the *twist angle*, and d_i is called the *offset distance at joint i*.

When the i-th joint is revolute (as in our case), then the parameters a_i, d_i, and α_i are constant while θ_i is a variable. For the indirect position problem, given the parameters a_i, d_i, and α_i, as well as one point, p, seen in the seventh coordinate system as p_7 and the first coordinate system as p_1 (joint i defines the i-th coordinate system), the goal is to determine the angles $\theta_1, \ldots, \theta_6$ that permit the hand of the robot to be located at position p. As derived in Manocha and Canny [84] and Tsai and Morgan [121], the system relating p_1 and p_7 is given in Equation 3–5, where both p_1 and p_7 are written in homogeneous coordinates [113] and A_i is the 4×4 matrix defined in Equation 3–6. This system can be reduced to a system of eight nonlinear equations in eight unknowns [84, 121].

$$p_1 = A_1 A_2 A_3 A_4 A_5 A_6 p_7 \tag{3–5}$$

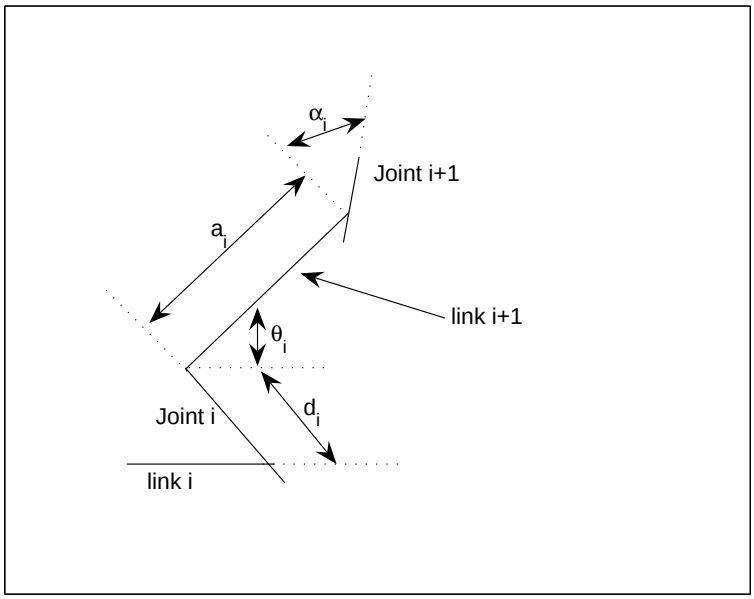

Figure 3–2: Robot arm parameters for link pair i and $i+1$.

$$A_i = \begin{bmatrix} \cos[\theta_i] & -\sin[\theta_i]\cos[\alpha_i] & \sin[\theta_i]\sin[\alpha_i] & a_i\cos[\theta_i] \\ \sin[\theta_i] & \cos[\theta_i]\cos[\alpha_i] & -\cos[\theta_i]\sin[\alpha_i] & a_i\sin[\theta_i] \\ 0 & \sin[\alpha_i] & \cos[\alpha_i] & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3–6}$$

When the hand position p and parameters a_i, d_i, and α_i are given the values as listed in Table 3–1, we get the system of Equations 3–7-3–14, where $x_1 = \cos[\theta_1]$, $x_2 = \sin[\theta_1]$, $x_3 = \cos[\theta_2]$, $x_4 = \sin[\theta_2]$, $x_5 = \cos[\theta_4]$, $x_6 = \sin[\theta_4]$, $x_7 = \cos[\theta_5]$, and $x_8 = \sin[\theta_5]$. It is easy to see that the domain for this problem is given by $x \in [-1,1]^8$. This is one of three instances specified by Tsai and Morgan [121], and the only instance specified by Floudas et al [35] and Kearfott [67]. In fact, in referring to this instance, Floudas et al. [35] state ‘‘this is a challenging problem.’’

Table 3–1: Indirect position problem parameter values.

i	a_i	d_i	α_i (degrees)	p
1	0.5000	0.1875	80.0	x: 0.22441776
2	1.0000	0.3750	15.0	y: 0.71549788
3	0.1250	0.2500	120.0	z: 0.79551628
4	0.6250	0.8750	75.0	
5	0.3125	0.5000	100.0	
6	0.2500	0.1250	60.0	

$$f_1(x) = 4.731 \cdot 10^{-3} x_1 x_3 - 0.3578 x_2 x_3 - 0.1238 x_1 + x_7 - 1.637 \cdot 10^{-3} x_2 - 0.9338 x_4 - 0.3571 = 0 \quad (3\text{–}7)$$

$$f_2(x) = 0.2238 x_1 x_3 + 0.7623 x_2 x_3 + 0.2638 x_1 - x_7 - 0.07745 x_2 - 0.6734 x_4 - 0.6022 = 0 \quad (3\text{–}8)$$

$$f_3(x) = x_6 x_8 + 0.3578 x_1 + 4.731 \cdot 10^{-3} x_2 = 0 \quad (3\text{–}9)$$

$$f_4(x) = -0.7623 x_1 + 0.2238 x_2 + 0.3461 = 0 \quad (3\text{–}10)$$

$$f_5(x) = x_1^2 + x_2^2 - 1 = 0 \quad (3\text{–}11)$$

$$f_6(x) = x_3^2 + x_4^2 - 1 = 0 \quad (3\text{–}12)$$

$$f_7(x) = x_5^2 + x_6^2 - 1 = 0 \quad (3\text{–}13)$$

$$f_8(x) = x_7^2 + x_8^2 - 1 = 0 \quad (3\text{–}14)$$

For this problem, we ran the enhanced C-GRASP algorithm 100 times (a different starting random seed for each run) with $\rho = 0.001$, $\beta = 1000$, and C-GRASP parameters $h_s = 0.5$, $h_e = 0.00001$, and $\rho_{lo} = 0.5$. In each case, C-GRASP was able to find all 16 known roots. The average CPU time needed to find all 16 roots was 75.36 seconds. Figure 3–3 displays the time as a function of the number of roots found.

3.3.2 Non-isothermal CSTR Steady-States

The next application we consider, originally described in Kubicek et al. [72], but also found in Floudas [33] and Floudas et al. [35], concerns a model of two continuous non-adiabatic stirred tank reactors. These reactors are in a series, at steady state with a

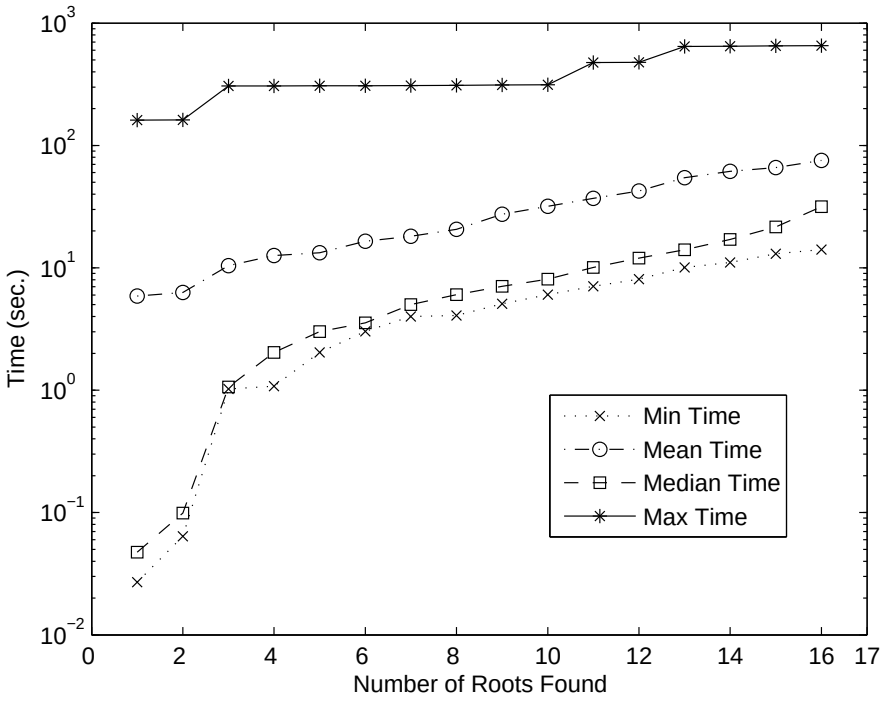

Figure 3–3: Time to find each root of the robot arm kinematics system (Equations 3–7-3–14).

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 in two unknowns (ϕ_1 and ϕ_2), seen in Equations 3–15-3–16. ϕ_1 and ϕ_2 represent the dimensionless temperatures in the two reactors, and are both bounded in the unit interval, i.e., $\phi_i \in [0,1]$ $i = 1,2$.

$$f_1 = (1-R)\left[\frac{D}{10(1+\beta_1)} - \phi_1\right]\exp\left(\frac{10\phi_1}{1+10\phi_1/\gamma}\right) - \phi_1 \tag{3–15}$$

$$f_2 = \phi_1 - (1+\beta_2)\phi_2 + (1-R)\cdot [D/10 - \beta_1\phi_1 - (1+\beta_2)\phi_2]\exp\left(\frac{10\phi_2}{1+10\phi_2/\gamma}\right) \tag{3–16}$$

Floudas et al. [35] 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.

Table 3–2: Average results for different recycle ratios, R.

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

As the recycle ratio parameter R takes on values in the set $\{0.935, 0.940, \ldots, 0.995\}$, the number of known solutions to this system varies between 1 and 7. For each value of the parameter R given in the set, we ran the C-GRASP heuristic 100 times, each time searching for all of the global minimizers corresponding to the solutions to the original system of Equations 3–15-3–16. Table 3–2 shows the results from these runs. The second column in the table lists the number of known roots for each R value. Columns three and four display the average number of roots found by C-GRASP and the time (in seconds) needed for C-GRASP to find those roots. As can be seen, the C-GRASP heuristic almost always finds all of the roots to the original system, while the computational time remains quite small.

3.3.3 Automotive Steering Mechanism

The next system we consider concerns the kinematic synthesis mechanism for automotive steering. This problem is originally considered in Pramanik [106], and is listed as a test problem in Merlot [92]. The Ackerman steering mechanism [127] 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

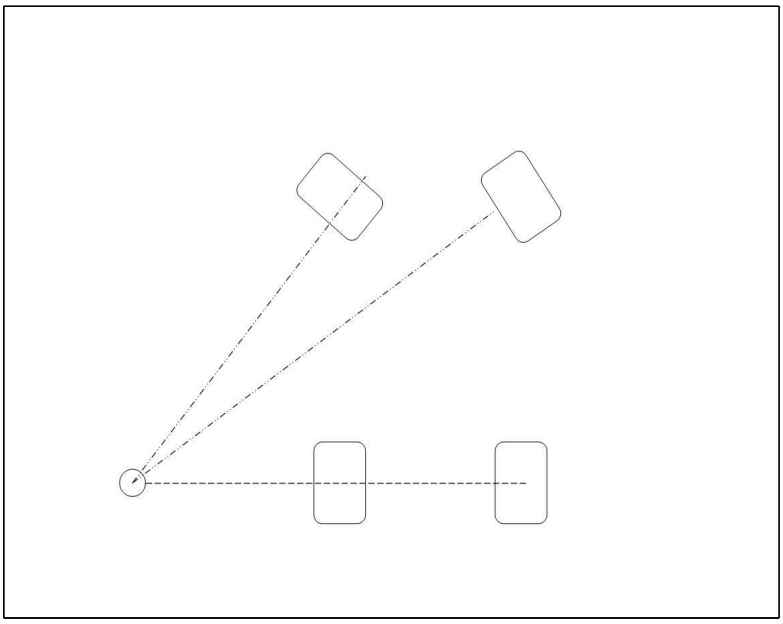

Figure 3–4: Steering turn maneuver (wheels are perpendicular to given line).

the wheels will have to be at different angles with respect to the non-steering wheels (see Figure 3–4).

The Ackerman design arranges the wheels automatically by moving the steering pivot inward. Pramanik [106] states that ‘‘the Ackerman design reveals progressive deviations from ideal steering with increasing ranges of motion.’’ Pramanik instead considers a six-member mechanism. This produces the system of Equations 3–17 for $i = 1,2,3$, where, E_i and F_i are defined by Equations 3–18 and 3–19, respectively. 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, ϕ_i, for $i = 0,1,2,3$.

$$G_i(\psi_i,\phi_i) = \Big[E_i(x_2\sin(\psi_i)-x_3)-F_i(x_2\sin(\phi_i)-x_3)\Big]^2+ \Big[F_i(1+x_2\cos(\phi_i))-E_i(x_2\cos(\psi_i)-1)\Big]^2- \Big[(1+x_2\cos(\phi_i))(x_2\sin(\psi_i)-x_3)x_1- (x_2\sin(\phi_i)-x_3)(x_2\cos(\psi_i)-x_3)x_1\Big]^2 \tag{3–17}$$

$$E_i = x_2(\cos(\phi_i)-\cos(\phi_0))-x_2x_3(\sin(\phi_i)-\sin(\phi_0))- (x_2\sin(\phi_i)-x_3)x_1 \tag{3–18}$$

$$F_i = -x_2\cos(\psi_i)-x_2x_3\sin(\psi_i)+x_2\cos(\psi_0)+x_1x_3+ (x_3-x_1)x_2\sin(\psi_0) \tag{3–19}$$

When the ψ_i and ϕ_i angles are given as in Table 3–3, there are two solutions to the system 3–17 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 to the system. The average time needed to find the first and second roots were 0.84 seconds and 5.06 seconds, respectively.

Table 3–3: Automotive steering mechanism angular parameters (in radians).

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

3.3.4 Additional Systems

The final three 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 [92]. The system is seen in Equations 3–20-3–21.

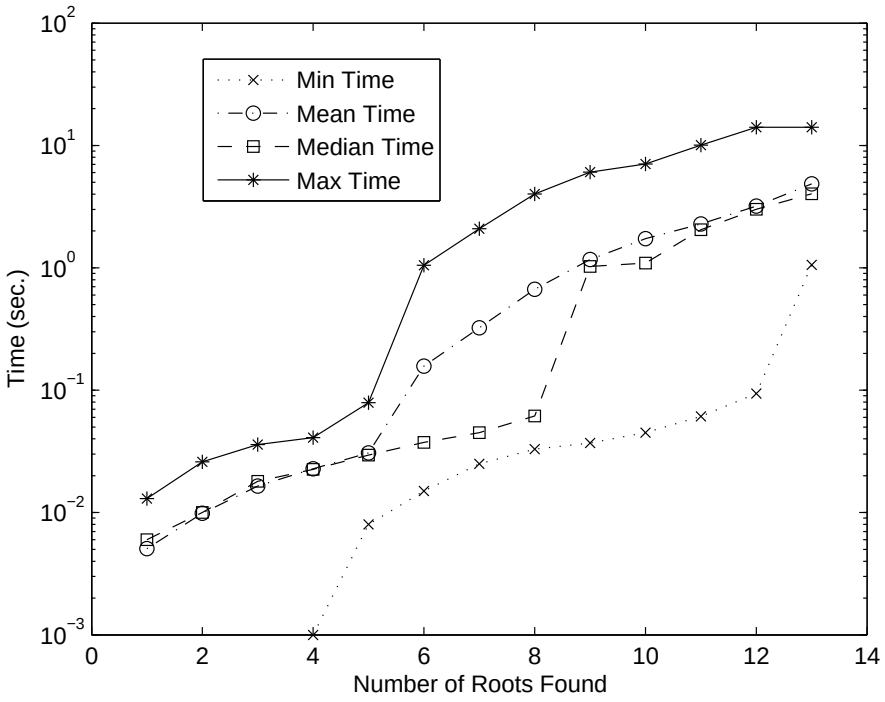

Figure 3–5: Time to find each root of the system (Equations 3–20-3–21).

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

$$f_1 = -\sin(x_1)\cos(x_2) - 2\cos(x_1)\sin(x_2) \tag{3–20}$$

$$f_2 = -\cos(x_1)\sin(x_2) - 2\sin(x_1)\cos(x_2) \tag{3–21}$$

We ran the C-GRASP algorithm 100 times on the corresponding optimization problem. In each run, C-GRASP was successful in finding all the roots. Figure 3–5 shows the time needed for the C-GRASP algorithm to find the roots to this system.

The second problem is a system of two nonlinear equations in two unknowns, found in Floudas et al. [35]. The system is given by Equations 3–22-3–23, where $x_1 \in [0.25, 1]$ and $x_2 \in [1.5, 2\pi]$. This system has two roots in the given domain. This system was converted into a global optimization problem and the C-GRASP heuristic was run 100 times. In each run, C-GRASP located both solutions, taking an average of 0.07 seconds to find the first

solution and 0.39 seconds to find the second solution.

$$f_1 = 0.5\sin(x_1x_2) - 0.25x_2/\pi - 0.5x_1 \tag{3–22}$$

$$f_2 = (1 - 0.25/\pi)(\exp(2x_1) - e)) + ex_2/\pi - 2ex_1 \tag{3–23}$$

The last system we look at is *Powell's singular function* [67, 98, 99]. This is a system of four nonlinear equations in four unknowns, Equations 3–24-3–27. Each of the variables is constrained to the interval $[-2,2]$. While this system only admits one solution in the domain, at the origin, Kearfott [67] remarks that ''The Jacobian matrix is null at the solution, and it poses a severe test of most methods.'' As with the previous problems, this system was converted into a global optimization problem and C-GRASP was run 100 times. In each run, C-GRASP was able to find the solution, and the average time needed to find this solution was 0.06 seconds.

$$f_1 = x_1 + 10x_2 \tag{3–24}$$

$$f_2 = \sqrt{5}(x_3 - x_4) \tag{3–25}$$

$$f_3 = (x_2 - 2x_3)^2 \tag{3–26}$$

$$f_4 = \sqrt{10}(x_1 - x_4)^2 \tag{3–27}$$

3.4 Conclusions

In this chapter, we have described a method to transform the problem of finding all solutions of a system of equations into one of finding all global minimizers of an optimization problem. As seen in Section 3.3, with this approach C-GRASP was able to find all of the solutions to the original systems almost all of the time. While it should be noted that no one heuristic will be able to solve all problems, the results in this chapter indicate that C-GRASP is a good heuristic for finding all solutions to systems of nonlinear equations.

CHAPTER 4
SENSOR REGISTRATION IN A SENSOR NETWORK

4.1 Introduction

In today's technology-driven environment, it is becoming more and more common for disparate sensors to view the same scene, or at least a partial overlap of the same scene. Military examples abound from the areas of missile defense [6, 75, 87], situation awareness [52, 53], and cooperating unmanned aerial vehicles (UAVs) [15]. Medical imaging and adverse drug reaction prediction [79, 80] are examples of two non-military areas where more than one sensor is receiving information of the same scene. In military situations, oftentimes the data each sensor infers from the scene is passed over communication links, either directly to other sensors viewing the scene, or to a central 'processor.' Hence, these sensors form a sensor network.

In either case (sensors communicating directly, or to a central processor), with multiple views of the same scene available, there lies the ability to gain a more precise representation of the scene than any one sensor could provide alone, by combining, or 'fusing', the information each sensor infers from its view. Examples include combining kinematic track states to reduce the uncertainty of the kinematic parameters [37, 56], adding an identification label to a kinematic track [53], and determining the number of objects of interest in the particular scene [6]. However, it is of the utmost importance that this fusion be done correctly.

One necessary pre-requisite for fusing data from multiple views of the same scene is to remove sensor registration[1] errors. These errors come in two forms: random and

[1] Sensor registration is also called data alignment, bias removal, and gridlock.

systematic. The random errors arise from detection and track processing techniques [7, 22], and at the fusion stage, these errors generally cannot be reduced. On the other hand, it is imperative that systematic errors be removed before data fusion occurs. The systematic errors arise from a number of sources, namely sensor calibration offset [51], platform flexure [51], sensor perspective offset [6, 7, 52, 53, 75, 87, 118], sensor internal clock errors [52, 53], and coordinate transformations [6, 7, 52, 53, 75, 87, 95]. Sensor registration is the process by which these systematic errors are removed. Without properly accounting for these systematic registration errors, the composite representation of the scene has the potential to be less precise then any one individual sensor's view, thus defeating the purpose of a sensor network [53].

Algorithms for sensor registration fall into two main categories. The first class determines the systematic error assuming the association of the data across the sensors is known. Techniques for this class include least-squares estimation [7, 22, 53, 118] and Kalman filtering [7, 37, 51]. The second class of algorithms does not assume knowledge of the data association. In effect, these types of algorithms attempt to determine the systematic error and the data correspondence at the same time [6, 7, 52, 68, 75, 87].

In this chapter, we apply the C-GRASP approach to determine systematic sensor registration errors between two sensors. We make no assumption on *a priori* knowledge of the correspondence of data between the sensors, thus this approach falls into the second class of sensor registration algorithms. This chapter is organized as follows. We begin, in Section 4.2, by deriving the optimization problem for sensor registration. Section 4.3 describes a decomposition approach for this problem, and how we incorporate C-GRASP into this decomposition. In Section 4.4, for a particular type of sensor registration problem, we compare the C-GRASP approach with two others, on simulated test cases. Conclusions are drawn in Section 4.5.

4.2 Sensor Registration Problem Derivation

In this section, we derive the sensor registration optimization problem. Assume we have two sensors, A and B, and the data from sensor B has been transformed into sensor A's coordinate system. Let N_A and N_B denote the number of targets seen by sensor A and B respectively. Without loss of generality, assume $N_A \leq N_B$. Denote by $P_A(i)$ and $C_A(i)$ the position and covariance estimates of the i-th track of sensor A, and similarly $P_B(j)$ and $C_B(j)$ for the j-th track of B. Also, there is an unknown systematic sensor registration error on the sensor B data, described by the function Ω (i.e., $\Omega(P_B(j))$, $\Omega(C_B(j))$ would remove the systematic error from the j-th track of sensor B). Then, the likelihood of associating the i-th track from sensor A with the j-th track of sensor B can be written, as a function of Ω, as given in Equation 4–1, where $t_{ij} = P_A(i) - \Omega(P_B(j))$ and $S_{ij} = C_A(i) + \Omega(C_B(j))$.

$$F_{ij}(\Omega) = \frac{1}{\sqrt{(2\pi)^m |S_{ij}|}} e^{-\frac{1}{2} t_{ij}^T S_{ij}^{-1} t_{ij}} \tag{4–1}$$

Now, for each track of A and B that represent the same object, say the i-th track of A and the j-th track of B, we would expect Ω to map $P_B(j)$ near $P_A(i)$. Thus making $-F_{ij}(\Omega)$ small (near -1). Generalizing this, it can be seen that the sensor registration problem can be written as the optimization problem given in Problem 4–2-4–6, where $\psi_{ij} = 1$ if the i-th track of A and the j-th track of B represent the same physical object and W defines the space of admissible transformations Ω (taken to be a linear space that maps the covariances homogeneously).

Problem 4–2-4–6 is a mixed-integer nonlinear programming problem. The binary variables are the ψ_{ij} and the continuous variables are encapsulated in the Ω function. Note that if Ω was known, then Problem 4–2-4–6 would be a linear assignment problem, while if the ψ_{ij} were known, then standard least squares techniques would determine Ω. However, in our case, neither the ψ_{ij} nor the Ω are known *a priori*. In fact, these are precisely the unknowns that we are trying to ascertain.

$$min - \sum_{i=1}^{N_A} \sum_{j=1}^{N_B} \psi_{ij} F_{ij}(\Omega) \tag{4–2}$$

$$\ni \sum_{i=1}^{N_A} \psi_{ij} \leq 1 \; \forall \; j \in \{1,\ldots,N_B\} \tag{4–3}$$

$$\sum_{j=1}^{N_B} \psi_{ij} \leq 1 \; \forall \; i \in \{1,\ldots,N_A\} \tag{4–4}$$

$$\psi_{ij} \in \{0,1\} \; \forall \; i \in \{1,\ldots,N_A\} \; \forall \; j \in \{1,\ldots,N_B\} \tag{4–5}$$

$$\Omega \in W \tag{4–6}$$

4.3 Problem Decomposition and Applying C-GRASP to Sensor Registration

Rather than apply C-GRASP to Problem 4–2-4–6 directly, we decompose the problem into two parts. The first step is to determine the best Ω. This is the step where C-GRASP is used. Then, with the best Ω found in the first step, the second step is to determine the assignment variables ψ_{ij}. Therefore, for the first step, we ignore the assignment variables, and solve the optimization problem given in Problem 4–7.

$$\min_{\Omega \in W} F(\Omega) = -\sum_{i=1}^{N_A} \sum_{j=1}^{N_B} F_{ij}(\Omega) \tag{4–7}$$

Problem 4–7 has the effect of finding the Ω the maps the most tracks of sensor B near tracks of sensor A. This decomposition approach can be seen as a consequence of Theorem 4.1.

Theorem 4.1. *Suppose the following:* ***i)*** *Ω preserves distances between points, i.e., $\|x-y\|_2 = \|\Omega(x)-\Omega(y)\|_2$.* ***ii)*** *In truth, there are N_c ($\geq dof(\Omega)$) tracks in common between sensor A and sensor B.* ***iii)*** *Track geometry is sufficiently random.*

Then, in the case where there is no random error (i.e. covariance matrices are $\mathbf{0}$), the Ω that minimizes $F(\Omega)$ in Problem 4–7 is precisely the Ω mapping the N_c tracks of sensor B onto their truthful corresponding tracks of sensor A.

Proof. Since Ω is a linear operator (operating homogeneously on covariance matrices), $C_B(j) \to 0 \Longrightarrow \Omega(C_B(j)) \to 0$. Also, $C_A(i), C_B(j) \to 0 \Longrightarrow S_{ij} = C_A(i) + \Omega(C_B(j)) \to 0$. Let $\hat{F}(\Omega)$ be the objective function of Problem 4–7, when no random error is present. Then,

$$\begin{aligned}
\hat{F}(\Omega) &= \lim_{\substack{C_A(i)\to 0\,\forall\, i \\ C_B(j)\to 0\,\forall\, j}} F(\Omega) \\
&= \lim_{\substack{C_A(i)\to 0\,\forall\, i \\ C_B(j)\to 0\,\forall\, j}} -\sum_{i=1}^{N_A}\sum_{j=1}^{N_B} F_{ij}(\Omega) \\
&= -\sum_{i=1}^{N_A}\sum_{j=1}^{N_B} \lim_{\substack{C_A(i)\to 0 \\ C_B(j)\to 0}} F_{ij}(\Omega) \\
&= -\sum_{i=1}^{N_A}\sum_{j=1}^{N_B} \lim_{\substack{C_A(i)\to 0 \\ C_B(j)\to 0}} \frac{1}{\sqrt{(2\pi)^m |S_{ij}|}} e^{-\frac{1}{2} t_{ij}^T S_{ij}^{-1} t_{ij}} \\
&= -\sum_{i=1}^{N_A}\sum_{j=1}^{N_B} \lim_{S_{ij}\to 0} \frac{1}{\sqrt{(2\pi)^m |S_{ij}|}} e^{-\frac{1}{2} t_{ij}^T S_{ij}^{-1} t_{ij}} \\
&= -\sum_{i=1}^{N_A}\sum_{j=1}^{N_B} \delta(\|t_{ij}\|^2)
\end{aligned}$$

where $\delta(x)$ is the Dirac-delta, or unit-impulse, function, i.e., $\delta(x-a) = 0$ if $x \neq a$ and $\int_{-\infty}^{\infty} \delta(x)\mathrm{dx} = 1$.

Now, $N_c \leq \min[N_A, N_B]$. Without loss of generality, assume that the first N_c tracks of sensor A correspond with the first N_c tracks of sensor B. For each $i \in \{1,\ldots,N_A\}$ and for each $j \in \{1,\ldots,N_B\}$, define Ω_{ij} by $P_A(i) = \Omega_{ij}(P_B(j))$, i.e., Ω_{ij} is the Ω that maps $P_B(j)$ into $P_A(i)$. Since there is no random error, we have $\bar{\Omega} = \Omega_{11} = \ldots = \Omega_{N_c N_c}$. Also, since the track geometry is random, $\mathbf{P}(\Omega_{ij} = \Omega_{i'j'}) = 0$ for the three cases:

Case a: $\forall\, i, i', j, j' \in \{1,\ldots,N_c\} \ni i \neq i'$;

Case b: $\forall\, i, i', j, j' \in \{1,\ldots,N_c\} \ni j \neq j'$;

Case c: $\forall\, i, i' \in \{N_c+1,\ldots,N_A\},\ j, j' \in \{N_c+1,\ldots,N_B\} \ni i \neq i' \vee j \neq j'$;

Therefore, for each $\Omega \neq \bar{\Omega}$, there exists (with probability 1) at most one i, j pair such that $\Omega = \Omega_{ij}$ and for $\Omega = \bar{\Omega}$, there exists N_c i, j pairs such that $\Omega = \Omega_{ij}$. Hence, the minimum of $\hat{F}(\Omega)$ will occur at $\Omega = \bar{\Omega}$. □

Once we find the best Ω by solving Problem 4–7, we can apply any linear assignment algorithm to determine the assignment of sensor A tracks to corrected sensor B tracks. In summary, the algorithm we propose to solve Problem 4–2-4–6 is given by:

1. Use C-GRASP to find the Ω that minimizes Problem 4–7.
2. Apply a linear assignment algorithm to determine the association between sensor A tracks and corrected sensor B tracks (we make use of the JVC linear assignment algorithm [65]).

4.4 Algorithm Comparison

To test our approach as outlined in Section 4.3, we chose the particular sensor registration application put forth in Blackman and Banh [6], Blackman and Popoli [7], Levedahl [75], and Maurer [87]. This application can be described as follows. We have two sensors, one active and one passive, both viewing a scene from possibly different perspectives. Each sensor, using no information from the other, creates tracks on the targets it senses. At some point in time, the active sensor sends its track data (state and covariance) via a communication link, to the passive sensor. The passive sensor transforms this track data into the local (i.e., passive) coordinate system. As a result of this coordinate transformation, as well as possible errors in the passive sensor perspective, systematic error is introduced into the active track data. The main component of this error is a translational shift, and other factors (e.g., rotation, scale, etc.) are small enough to be ignored. Therefore, for this problem, Ω defines an unknown translation, such that when Ω is applied to the active sensor data, the systematic error is removed. The goal is to correct for the translational error, and at the same time determine the assignment of active tracks to passive tracks. Figure 4–1 shows a simple example of 3 passive tracks (triangles) and 5 active tracks (squares) when systematic error is present (the covariances have been omitted from the figure to reduce clutter).

We compare our C-GRASP based approach against the two approaches found in Blackman and Banh [6] and Levedahl [75] (from here on denoted as the Blackman and

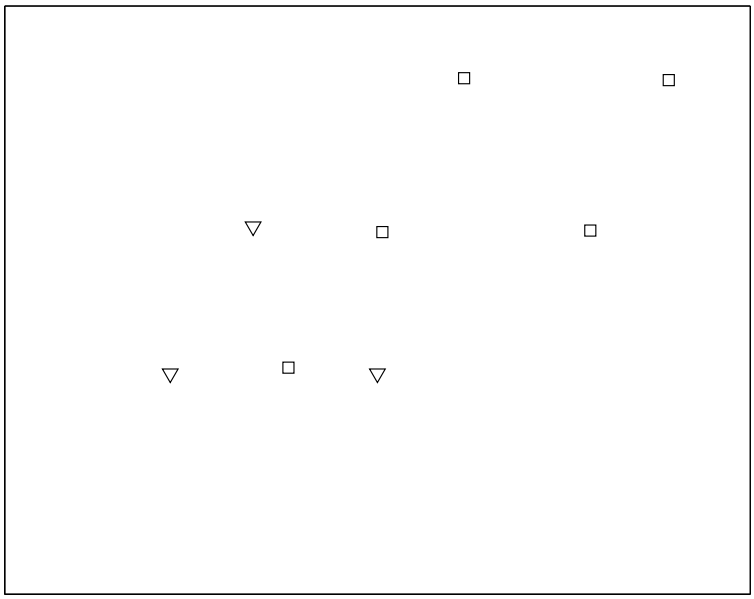

Figure 4–1: Passive sensor image plane. (Triangles represent passive tracks. Squares represent active tracks. Systematic error present).

Levedahl algorithms, respectively). The Blackman approach is an iterative scheme that starts with an initial estimate of the systematic error, applies a linear assignment algorithm (with gating) to the current corrected scene, to produce a set of assignments. From these assignments, a new estimate of the systematic error is determined. This process continues until the systematic error estimate achieves convergence (i.e., the difference between the previous and current estimate is small). The Levedahl algorithm looks at each set of possible assignments, with gating involved to limit the set of feasible assignments. For each feasible assignment vector, an estimate of the systematic error is computed. The assignment and systematic error that produces the maximum of an objective function (similar to the objective function in Problem 4–7) is returned as the answer. N.B.: In the worst-case scenario, where gating does not eliminate any of the feasible assignments, the Levedahl algorithm will need to examine $\sum_{k=0}^{m} \binom{m}{k} \frac{n!}{(n-k)!}$ different assignment vectors, where m and n are the number of targets seen by the passive and active sensor, respectively.

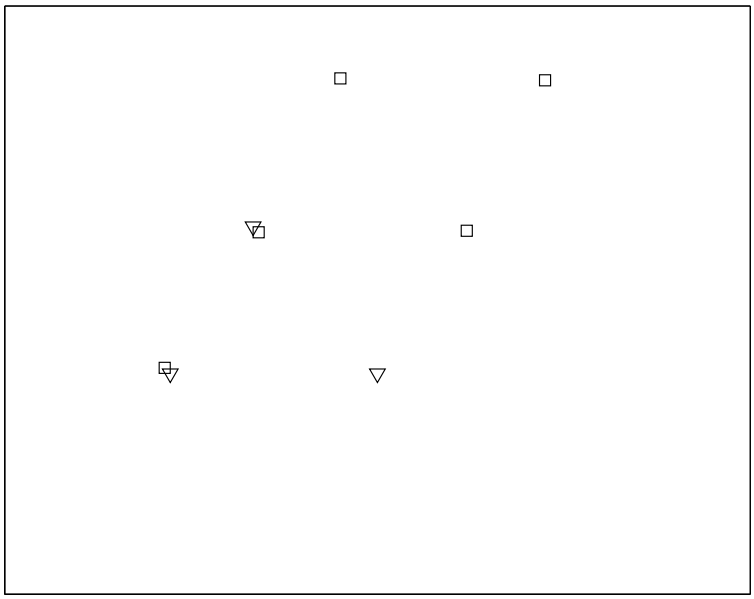

Figure 4–2: Blackman bias removal algorithm results when applied to the sensor registration problem defined in Figure 4–1. Systematic error has not been removed correctly.

Both the Blackman and Levedahl algorithms were implemented by the author in the C++ programming language. The same hardware and compiler options were utilized for these implementations as was described for the C-GRASP algorithms.

As a simple example, consider again the scenario depicted in Figure 4–1. Applying the Blackman algorithm to this scenario, with an initial guess of $\mathbf{0}$ for the translational bias, produces the result seen in Figure 4–2. As is clear from the figure, the systematic bias has not been removed correctly. Figure 4–3 shows the result of applying the Levedahl algorithm to this scenario. The Levedahl algorithm correctly removes the systematic error present in the scene.

Figure 4–4 shows the objective function, $F(\Omega)$, from Problem 4–7, for the scenario depicted in Figure 4–1. From this figure, it is evident that there are 10 local minima, with only one of those being the global minimum. This global minimum solution corresponds precisely with the translational bias between the active and passive tracks. Finally, Figure 4–5

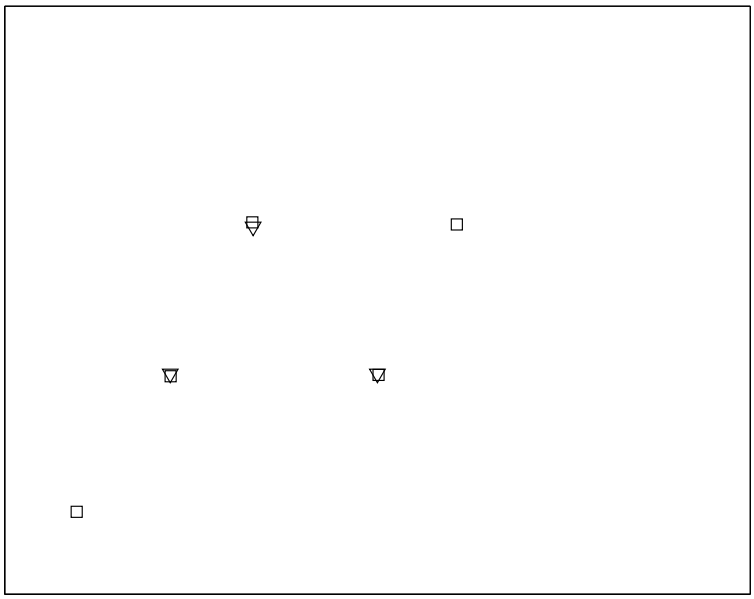

Figure 4–3: Levedahl bias removal algorithm results when applied to the sensor registration problem defined in Figure 4–1. Systematic error has been removed correctly.

shows the result upon running the C-GRASP algorithm to remove the systematic error; the three passive tracks align with the correct active tracks.

To compare the three different approaches, we look at a few different scenario classes. Without loss of generality, let sensor A be the passive sensor and sensor B the active sensor. We define a test class as a vector $[N_A, N_B, N_c, S_A]$, where N_A denotes the number of targets seen by sensor A, N_B denotes the number of targets seen by sensor B, N_c defines the number of targets in common to the two sensors, and S_A controls the maximum 1-σ covariance value for all targets as seen by sensor A. For each test class, we created 100 random scenarios, choosing the target positions in a square of length 20 km, and fixing the maximum 1-σ covariance value for all targets as seen by sensor B to be 3 km. After creating each scenario, we applied a random translational bias to all sensor B target data. The goal of the three algorithms is thus to determine the truthful bias, or an approximation to this bias, as well as the correct assignment of passive to active tracks. Table 4–1 lists the

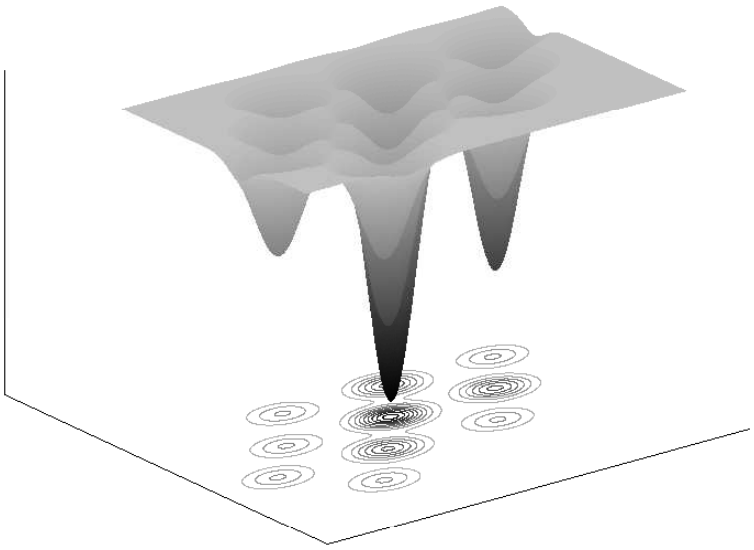

Figure 4–4: Plot of objective function, $F(\Omega)$, for C-GRASP approach to sensor registration.

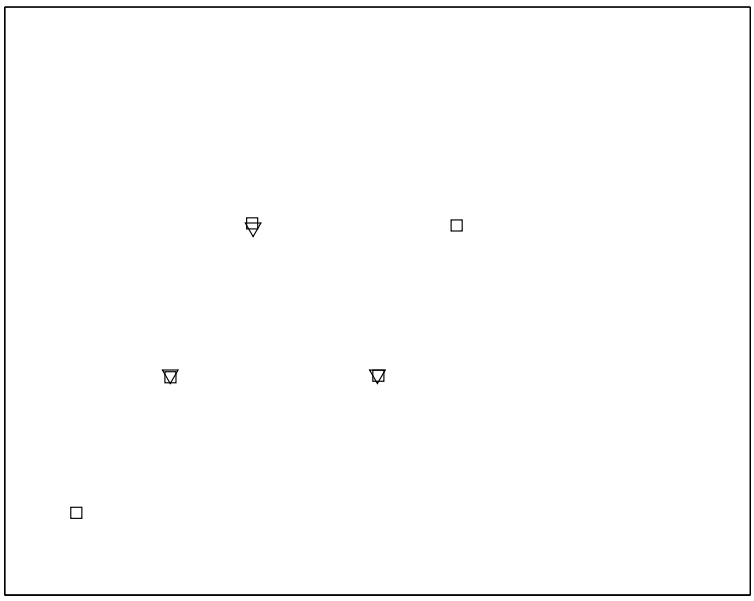

Figure 4–5: Enhanced C-GRASP bias removal algorithm results when applied to the sensor registration problem defined in Figure 4–1. Systematic error has been removed correctly.

Table 4–1: Sensor registration test classes.

Class	N_A	N_B	N_c	S_A
1	4	6	2	$\{0.5, 1.0, 2.0, 3.0\}$
2	4	6	3	$\{0.5, 1.0, 2.0, 3.0\}$
3	4	6	4	$\{0.5, 1.0, 2.0, 3.0\}$
4	5	10	2	$\{0.5, 1.0, 2.0, 3.0\}$
5	5	10	3	$\{0.5, 1.0, 2.0, 3.0\}$
6	5	10	4	$\{0.5, 1.0, 2.0, 3.0\}$
7	5	10	5	$\{0.5, 1.0, 2.0, 3.0\}$
8	7	20	2	$\{0.5, 1.0, 2.0, 3.0\}$
9	7	20	3	$\{0.5, 1.0, 2.0, 3.0\}$
10	7	20	4	$\{0.5, 1.0, 2.0, 3.0\}$
11	7	20	5	$\{0.5, 1.0, 2.0, 3.0\}$
12	7	20	6	$\{0.5, 1.0, 2.0, 3.0\}$
13	7	20	7	$\{0.5, 1.0, 2.0, 3.0\}$

test classes examined in this paper. To measure the performance of the three algorithms, we looked at two metrics: the percentage of correct assignments, and the distance between each truthful correspondence. These two metrics were computed for each scenario, and averaged over each test class.

Tables 4–2 to 4–5 display the results for each test class. For each table, the first three columns show the average percentage of correct assignments for the three algorithms. Columns 4 to 6 show the average distance between truthful correspondences, after the bias has been removed. The underlined number in each row of each table denotes the best of the three algorithms. As is clearly seen from these tables, the C-GRASP approach does better than the other two algorithms with respect to percentage of correct assignments. For the average distance metric, the Blackman algorithm almost always performs the worst, while the Levedahl and C-GRASP approaches perform about the same. These tables, along with knowledge of the Levedahl upper-bound on the number of possible assignments to examine, lend support to C-GRASP comparing favorably to both the Blackman and Levedahl approaches.

Table 4–2: Sensor registration results with $\mathbf{S_A} = 0.5$ km.

Test	Avg. Correct Assign			Avg. Distance		
Class	Blackman	Levedahl	C-GRASP	Blackman	Levedahl	C-GRASP
1	0.530	0.715	0.655	2.294	1.727	1.950
2	0.820	0.917	0.943	0.810	0.286	0.318
3	0.910	0.965	0.978	0.400	0.059	0.063
4	0.465	0.570	0.585	2.441	2.399	2.352
5	0.637	0.813	0.800	1.621	1.048	1.150
6	0.798	0.880	0.913	0.785	0.506	0.523
7	0.836	0.930	0.962	0.692	0.118	0.114
8	0.325	0.380	0.450	2.553	2.630	2.458
9	0.493	0.540	0.520	2.023	2.027	2.445
10	0.553	0.668	0.718	1.793	1.532	1.564
11	0.652	0.780	0.874	1.393	0.683	0.685
12	0.705	0.872	0.900	1.082	0.322	0.299
13	0.799	0.899	0.951	0.910	0.111	0.136

Table 4–3: Sensor registration results with $\mathbf{S_A} = 1.0$ km.

Test	Avg. Correct Assign			Avg. Distance		
Class	Blackman	Levedahl	C-GRASP	Blackman	Levedahl	C-GRASP
1	0.625	0.770	0.785	1.826	1.336	1.305
2	0.793	0.880	0.927	1.043	0.798	0.497
3	0.945	0.960	0.993	0.282	0.061	0.092
4	0.450	0.650	0.615	2.607	2.090	2.297
5	0.677	0.853	0.833	1.450	0.802	0.901
6	0.820	0.923	0.943	0.826	0.348	0.342
7	0.856	0.948	0.972	0.677	0.058	0.024
8	0.375	0.345	0.430	2.342	2.760	2.510
9	0.483	0.617	0.653	1.995	1.778	1.548
10	0.573	0.755	0.757	1.783	0.946	1.003
11	0.594	0.802	0.826	1.488	0.724	0.732
12	0.745	0.863	0.952	1.026	0.381	0.298
13	0.747	0.920	0.987	0.995	0.095	0.096

Table 4–4: Sensor registration results with $\mathbf{S_A} = 2.0$ km.

Test	Avg. Correct Assign			Avg. Distance		
Class	Blackman	Levedahl	C-GRASP	Blackman	Levedahl	C-GRASP
1	0.600	0.745	0.830	1.888	1.658	1.046
2	0.850	0.943	0.957	0.617	0.248	0.258
3	0.975	0.968	0.965	0.151	0.042	0.034
4	0.560	0.465	0.500	1.712	2.983	2.558
5	0.680	0.830	0.817	1.320	0.884	0.900
6	0.818	0.905	0.940	0.797	0.298	0.386
7	0.870	0.946	0.986	0.727	0.085	0.091
8	0.340	0.360	0.405	2.687	2.915	2.885
9	0.463	0.530	0.570	2.100	2.128	2.032
10	0.493	0.750	0.763	2.066	0.882	0.781
11	0.638	0.800	0.826	1.419	0.709	0.724
12	0.732	0.878	0.945	1.190	0.341	0.355
13	0.766	0.930	0.959	1.137	0.125	0.137

Table 4–5: Sensor registration results with $\mathbf{S_A} = 3.0$ km.

Test	Avg. Correct Assign			Avg. Distance		
Class	Blackman	Levedahl	C-GRASP	Blackman	Levedahl	C-GRASP
1	0.615	0.745	0.710	1.948	1.476	1.796
2	0.833	0.910	0.920	0.696	0.436	0.449
3	0.973	0.973	0.983	0.117	0.032	0.031
4	0.495	0.495	0.655	2.309	2.452	1.752
5	0.653	0.760	0.763	1.642	1.227	1.230
6	0.783	0.935	0.898	0.891	0.275	0.253
7	0.858	0.962	0.966	0.719	0.055	0.061
8	0.345	0.235	0.370	2.695	3.610	2.887
9	0.447	0.487	0.550	2.242	2.066	2.068
10	0.563	0.680	0.663	1.789	1.378	1.344
11	0.622	0.790	0.832	1.608	0.753	0.750
12	0.740	0.863	0.898	1.014	0.348	0.341
13	0.716	0.944	0.961	1.220	0.092	0.093

4.5 Conclusions

In this chapter, we have applied the C-GRASP algorithm to the problem of registering sensors in a sensor network. We have shown that our approach performs better than the approaches of Blackman [6] and Levedahl [75]. In addition, the C-GRASP algorithm does not suffer from Levedahl's exponential worst-case performance bound. Hence, our approach can be seen as an attractive algorithm for the removal of systematic sensor registration error. Also, it should be noted that the Blackman approach does not generalize to the case where the registration is a more general transformation.

Since both the Blackman and Levedahl algorithms have been implemented in fielded military systems, it seems natural that the C-GRASP approach will replace these algorithms in future releases of these fielded systems. Future work involves testing this C-GRASP approach on real track geometries, rather than the random scenarios presented here.

CHAPTER 5
DETERMINATION OF DRUGS RESPONSIBLE FOR ADVERSE REACTIONS

5.1 Introduction

An adverse drug reaction (ADR) is the onset of an untoward medical condition caused by the administration of a therapeutic drug [46]. 5% of hospital admissions, 28% of emergency room visits, and 5% of hospital deaths can be attributed to ADRs [128]. ADRs are often not recognized during the testing phase of a new drug, as these testing studies are generally small and of short duration. Thus, most ADRs are detected as a result of post-marketing studies, i.e., when the drug is available and prescribed to the general public. The U.S. Food and Drug Administration (FDA), The World Health Organization (WHO), as well as other countries and organizations, have created databases specifically for storing ADRs reported by practicing physicians and hospitals [3, 11, 122, 123].

The adverse drug reaction problem can be set up as follows: There are n patients, m drugs, and c possible reaction classes. Each of the n patients has taken a known subset of the m drugs. In addition, each patient is exhibiting symptoms from a subset of the c possible reactions. The goal is to determine a weight, or 'probability', for each drug being the cause of each reaction. There have been many proposed solutions for the ADR problem based on optimization techniques [2, 46, 64, 79, 80, 81, 82, 83, 110, 128, 131]. The approach of Mammadov et al. [81, 82, 83] (referred to as the Mammadov Algorithm) sets up the ADR problem as a nonlinear optimization problem subject to bound constraints.

In this chapter, we present a modification to the Mammadov et al. [81, 82, 83] formulation. We solve this formulation using the enhanced version of C-GRASP (with some minor changes). We also examine the Average Precision metric, the metric used in Mammadov et al. [81, 82, 83] for performance analysis. This chapter is organized as

follows. We begin, in Section 5.2, by describing the Australian adverse drug reaction data. Section 5.3 is devoted to introducing both the Mammadov algorithm and our modifications. We compare these two formulations in Section 5.4. Section 5.5 provides some concluding remarks.

5.2 Australian Adverse Drug Reaction Data

The aim of the Australian Adverse Drug Reaction Advisory Committee (ADRAC) is to detect signals from adverse drug reactions as early as possible [3]. The ADRAC data contains 137,297 records collected from 1971 to 2001. There exist 18 system organ reaction classes, one of those being the cardiovascular class. The cardiovascular reaction class is broken down into 4 subclasses. We limit our study to data that has at least one reaction from the cardiovascular class. Therefore, for each record of data, there are five possible reactions to consider, four from the cardiovascular class, and a fifth reaction that encompasses all reactions that are not part of the cardiovascular class.

For the data we received[1] , each record corresponds with a patient. Each record contains 26 fields. The first 10 fields represent up to 10 drugs that the patient was prescribed. Each drug is represented as a unique number. If a patient took less than 10 drugs, then the remaining drug fields are set to 0. Fields 11 through 15 contain a binary vector representing which reactions were observed for the patient. Field 16 contains the year that this ADR was written. And fields 17 through 26 contain a binary vector representing which of the drugs in the first 10 fields are suspected of causing the observed reactions, in the opinion of the doctor writing the ADR. As an example, consider the following record:

1984, 4871, 5544, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1972, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0.

This adverse drug report was written in 1972. The patient was prescribed drugs 1984, 4871,

[1] Data was received from Musa Mammadov on April 6, 2006 in private email communication.

and 5544. Reactions 1, 4, and 5 occurred. The doctor writing the report was of the opinion that drug 4871 caused the observed reactions.

5.3 ADR Problem Formulation

We begin this section by presenting the Mammadov algorithm (both the problem formulation and the solution technique). Then, we present our modification to the formulation, as well as some minor changes to the enhanced C-GRASP algorithm. However, first we introduce some essential notation.

We define n to be the number of patients in the data set, m to be the total number of drugs over all patients, and c to be the number of possible reactions. Let X^a be the $n \times m$ binary matrix defined in Equation 5–1, where drug j refers to the j-th drug in the drug list for this data set. The matrix X^a corresponds to the patient/drug information if we consider all prescribed drugs for each patient. In a similar way, we can define X^s (see Equation 5–2) to represent the patient/drug information if we only consider the suspect drugs for each patient. N.B.: $X^s_{ij} = 1 \Longrightarrow X^a_{ij} = 1$.

$$X^a_{i,j} = \begin{cases} 1 & \text{if drug } j \text{ was prescribed for patient } i \\ 0 & o.w. \end{cases} \tag{5–1}$$

$$X^s_{i,j} = \begin{cases} 1 & \text{if drug } j \text{ is a suspect drug for patient } i \\ 0 & o.w. \end{cases} \tag{5–2}$$

We let Y be the $n \times c$ binary matrix defined in Equation 5–3, representing which reactions were observed for each patient.

$$Y_{i,k} = \begin{cases} 1 & \text{if reaction } k \text{ was observed for patient } i \\ 0 & o.w. \end{cases} \tag{5–3}$$

We are tasked with finding a matrix W, of size $m \times c$, where $W_{j,k}$ is the weight, or 'probability', of drug j causing reaction k.

The objective function defined in the Mammadov algorithm, $F(W;\ p,\ X)$, is presented in Equation 5–4, where $p \in \mathbb{Z}$ and $X \in \{X^a, X^s\}$.

$$F(W;\ p,\ X) = \frac{1}{n}\sum_{i=1}^{n}\left[\|Y_i\|_1^{-p} \cdot \|\frac{\|Y_i\|_1}{\|X_iW\|_1}(X_iW) - Y_i\|_2^2\right] \tag{5–4}$$

The optimization problem in the Mammadov algorithm is then to minimize $F(W;\ p,\ X)$ such that $0 \leq W \leq 1$. However, since the size of the ADR problem can be very large, the Mammadov algorithm does not solve this problem; rather it solves an approximation to this problem, discussed presently.

For drug j, define $X[j]$ to be the set of all patients that were only prescribed drug j, i.e., $X[j] = \{i \in \{1,\ldots,n\} \mid X_{i,j} = 1 \wedge X_{i,j'} = 0\ \forall\ j' \neq j\}$. In a similar way, define $X_{all}[j]$ to be the set of patients that were prescribed drug j. It is clear that $X[j] \subseteq X_{all}[j]$. The solution technique taken in the Mammadov algorithm is to compute $W_{j,k}$ via Equation 5–5[2] , where $W^*_{j,k}$ and $W^{**}_{j,k}$ are defined by equations 5–6 and 5–7 respectively, $\Delta''[i]$ is the set of all drugs prescribed to patient i whose weights are not calculated using Equation 5–6, and $\texttt{rem}(Y_{i,k})$ can be seen as the remainder of the reaction available after accounting for the drugs j whose weights are computed using Equation 5–6.

$$W_{j,k} = \begin{cases} W^*_{j,k} & \text{if } a|X[j]| + 100b\frac{|X[j]|}{|X_{all}[j]|} \geq \rho^* \\ W^{**}_{j,k} & o.w. \end{cases} \tag{5–5}$$

$$W^*_{j,k} = \left[\sum_{i\in X[j]} \|Y_i\|^{2-p}\right]^{-1} \cdot \left[\sum_{i\in X[j]} \|Y_i\|^{1-p} Y_{i,k}\right] \tag{5–6}$$

$$W^{**}_{j,k} = \left[\sum_{i\in X_{all}[j]} \|Y_i\|^{2-p}\right]^{-1} \cdot \left[\sum_{i\in X_{all}[j]} \|Y_i\|^{1-p} \frac{\texttt{rem}(Y_{i,k})}{|\Delta''[i]|}\right] \tag{5–7}$$

As stated above, the Mammadov formulation allows X to be either X^a or X^s. Our modification to this problem formulation is to consider $X = \beta X^a + (1-\beta)X^s$, where $\beta \in$

[2] In all experiments to follow, a, b, and ρ^* were set to 1, 1, and 80, respectively.

$[0,1]$ is unknown. Thus, X becomes a convex combination of all drugs and suspected drugs for each patient. Looked at in another way, this definition of X can be seen as providing a discount measure to the non-suspected drugs, as, for each patient, the suspect drugs are found in both X_i^a and X_i^s. Our formulation, given in Equation 5–8, is a minimization problem solving for W, as well as β.

$$\min_{W,\beta} F(W;\ p,\ X)\ \ni\ 0 \leq W, \beta \leq 1 \tag{5–8}$$

Since we only have box constraints for the formulation given by 5–8, we employ the enhanced version of C-GRASP, with some minor changes. Rather than allowing all variables to start as 'un-fixed' in the construction procedure, we limit the number of starting un-fixed variables to 10, chosen randomly each time the construction procedure is called. Had this change not been implemented, it was estimated that each call to the construction procedure would have taken more than one year of computing time. We also run the C-GRASP algorithm for a fixed number $(2,000)$ of construction/local improvement iterations, keeping the discretization parameter fixed at $h = 0.1$. As will be seen in the Section 5.4, even with these restrictions to the enhanced C-GRASP algorithm, we are still able to find a better solution than the Mammadov algorithm.

5.4 Metrics and Results

In this section, we will present some metrics used to evaluate the solutions found for the ADR problem. We will then discuss some characteristics of the particular data, and compare the Mammadov algorithm against our approach using the metrics.

5.4.1 ADR Metrics

Mammadov et al. [81, 82, 83] defines only one metric to measure the performance of their algorithm, the Average Precision metric. In this section, we will describe the average precision metric, present an example detailing its computation, and discuss a potential pitfall of this metric. We will also suggest two more conventional metrics that have been used extensively in the literature to measure distance between vectors.

Suppose we have a solution W to the ADR optimization problem. We would like some way of measuring the quality of this solution. For each patient i, define the computed reaction of this patient to be $\hat{Y}_i = X_i W$. Recall from Section 5.3 that the true, or observed, reaction for this patient is given by Y_i. Let $\mathbb{T}_i$ be the set of all permutations of the indices $1,\ldots,c$ such that the permutation applied to $\hat{Y}_i$ produces a non-increasing sequence. Then, for each $\tau \in \mathbb{T}_i$, the precision for τ, $P_\tau(X_i)$, is defined in Equation 5–9, with $R_{i,k}(\tau)$ and $R'_{i,k}(\tau)$ defined in Equations 5–10 and 5–11, respectively.

$$P_\tau(X_i) = \frac{1}{\|Y_i\|_1} \sum_{k:Y_{i,k}=1} \frac{|R'_{i,k}(\tau)|}{|R_{i,k}(\tau)|} \tag{5–9}$$

$$R_{i,k}(\tau) = \{\ell \in \{1,\ldots,c\} \mid \ell \text{ is listed before } k \text{ in } \tau\} \cup \{k\} \tag{5–10}$$

$$R'_{i,k}(\tau) = \{\ell \in \{1,\ldots,c\} \mid Y_{i,\ell} = 1 \ \wedge\ \ell \text{ is listed before } k \text{ in } \tau\} \cup \{k\} \tag{5–11}$$

The average precision for patient i, $P(X_i)$, is then defined to be the average of the best and worst precisions over all $\tau \in \mathbb{T}_i$. The average precision is given as the average over the average precision of each patient, i.e., $P_{AP} = \frac{1}{n}\sum_{i=1}^{n} P(X_i)$.

As an example of computing the precision for a patient, suppose that for patient i, the observed reaction is $Y_i = \{1,0,1,1,0\}$ and the computed reaction is found to be $\hat{Y}_i = \{0.9,0.1,0.1,0.9,0.1\}$. The set of all index permutations $\mathbb{T}_i$ that produce a non-increasing sequence for the elements of $\hat{Y}_i$ is listed in Table 5–1. Looking at the permutation $\tau = \{1,4,5,3,2\}$, we see that $R_{i,1} = R'_{i,1} = \{1\}$, $R_{i,4} = R'_{i,4} = \{1,4\}$, $R_{i,3} = \{1,4,5,3\}$, and $R'_{i,3} = \{1,4,3\}$. Therefore, $P_\tau(X_i) \approx 0.9167$. The best precision for this patient (from permutations that place 3 before 2 and 5) is 1, while the worst precision for this patient (from permutations that place 3 after 2 and 5) is approximately 0.8667. Hence, this patient would have an average precision of $P(X_i) \approx 0.9333$.

We now provide an example highlighting a potential shortcoming of the average precision metric. Consider the same patient observed reaction, namely $Y_i = \{1,0,1,1,0\}$,

Table 5–1: Permutations of $\{1,2,3,4,5\}$ that are elements of $\mathbb{T}_i$.

$\{1,4,2,3,5\}$	$\{4,1,2,3,5\}$
$\{1,4,2,5,3\}$	$\{4,1,2,5,3\}$
$\{1,4,3,2,5\}$	$\{4,1,3,2,5\}$
$\{1,4,3,5,2\}$	$\{4,1,3,5,2\}$
$\{1,4,5,2,3\}$	$\{4,1,5,2,3\}$
$\{1,4,5,3,2\}$	$\{4,1,5,3,2\}$

and suppose one approach to solving the ADR problem produces a computed reaction of $\hat{Y}_i' = \{0.1, 0.01, 0.07, 0.04, 0.039\}$, while a second approach produces the computed reaction of $\hat{Y}_i'' = \{0.9, 0.01, 0.7, 0.8, 0.039\}$. While it is easy to see that both of these computed reactions will produce an average precision of 1.0 for this patient, obviously the second computed reaction, $\hat{Y}_i''$, predicts much more closely the true observed reaction then does the first. Hence, for this particular example, it is unclear what the average precision metric is really measuring.

To aid in measuring how well the different solution approaches are performing, in addition to using the average precision metric we also look at the standard least squares residual, $LS(W)$ (defined in Equation 5–12) and the average maximum deviation, $AMD(W)$, (defined in Equation 5–13). Using these conventional metrics will lend stronger support to one algorithm performing better than another.

$$LS(W) = \frac{1}{n}\sum_{i=1}^{n} \|\hat{Y}_i - Y_i\|_2^2 \tag{5–12}$$

$$AMD(W) = \frac{1}{n}\sum_{i=1}^{n} \max_{k\in\{1,\ldots,c\}} [|\hat{Y}_i - Y_i|] \tag{5–13}$$

5.4.2 Scenarios and Results

In this section, we explain the testing scenarios and compare the results produced using our approach with those of the Mammadov algorithm.

The ADRAC data set contains data for the years 1972 to 2001 inclusive. The approach taken by Mammadov et al. [81, 82, 83], which we adhere to, was to consider the years 1996 through 2001. For each of these years, a training data set was formed using the data from

Table 5–2: Drug reaction data characteristics.

	Training Data			Testing Data
Year	# Patients	# Drugs	# Variables	# Patients
1996	12,600	2,253	11,265	1,049
1997	13,747	2,375	11,875	1,091
1998	15,001	2,497	12,485	1,418
1999	16,684	2,661	13,305	1,746
2000	18,599	2,786	13,930	1,988
2001	20,745	2,916	14,580	1,054

Table 5–3: Average precision comparison.

	Mammadov				C-GRASP	
	All		Suspect			
Year	Train	Test	Train	Test	Train	Test
1996	0.8145	0.7997	0.8291	0.8025	0.8780	0.8517
1997	0.8153	0.7905	0.8280	0.7979	0.8779	0.8528
1998	0.8158	0.7786	0.8282	0.7802	0.8758	0.8569
1999	0.8152	0.8041	0.8272	0.8091	0.8777	0.8727
2000	0.8169	0.7757	0.8272	0.7788	0.8755	0.8380
2001	0.8135	0.7687	0.8232	0.7685	0.8722	0.8283

year 1972 to the previous year. This is the data that will be used to determine the weight matrix W. Once the weight matrix is determined, it is applied to the current year data to measure how well W predicts the true reactions. As an example, for the year 1998, data from years 1972 to 1997 (the training data) is used in the solution approaches to determine W. Then, W is applied to the year 1998 data (the test data) to measure the performance of W. In addition, it is also useful to measure the performance of W on the training data. Table 5–2 lists characteristics for each of the training and test data sets.

Tables 5–3-5–5 display the performance of both the Mammadov algorithm and our enhanced C-GRASP algorithm for the three metrics (N.B.: The Mammadov algorithm was coded by the author in Matlab in order to verify the average precision results, as well as determine values for the least squares and average maximum deviation metrics).

As is clear from these tables, overall the C-GRASP approach performs better for both the training and testing data for each year, when looking at all three metrics. The Mammadov algorithm, by not considering all of the data when computing W and not forming a convex combination of all drugs and suspect drugs, appears to be limited in

Table 5–4: Least squares residual comparison.

	Mammadov				C-GRASP	
	All		Suspect			
Year	Train	Test	Train	Test	Train	Test
1996	0.0091	0.0330	0.0089	0.0355	0.0090	0.0345
1997	0.0088	0.0321	0.0085	0.0352	0.0087	0.0344
1998	0.0084	0.0264	0.0082	0.0297	0.0085	0.0310
1999	0.0079	0.0245	0.0077	0.0245	0.0080	0.0245
2000	0.0075	0.0239	0.0073	0.0232	0.0076	0.0230
2001	0.0071	0.0331	0.0069	0.0372	0.0074	0.0348

Table 5–5: Average maximum deviation comparison.

	Mammadov				C-GRASP	
	All		Suspect			
Year	Train	Test	Train	Test	Train	Test
1996	0.7551	0.7869	0.7306	0.8138	0.6977	0.7836
1997	0.7572	0.7653	0.7321	0.8196	0.7058	0.7719
1998	0.7525	0.7486	0.7321	0.7994	0.7140	0.7802
1999	0.7497	0.7569	0.7296	0.7484	0.7124	0.7305
2000	0.7464	0.7877	0.7267	0.7588	0.7081	0.6950
2001	0.7469	0.7902	0.7284	0.8521	0.7255	0.7544

its ability to achieve a solution close to the global minimum. The C-GRASP algorithm does not have this limitation. Of note is that, for each year, each iteration of the C-GRASP algorithm improved upon the previous iteration. Therefore, letting the C-GRASP algorithm run for more than $2,000$ iterations should produce an even better solution. Table 5–6 displays the β values (discounting values) found by the C-GRASP approach for each year.

5.5 Conclusions

This chapter has seen us apply the enhanced C-GRASP algorithm to a problem concerning adverse drug reactions. Depending on the data, the number of variables in the optimization formulation can be quite large. We have discussed the approach taken by

Table 5–6: Discount factor (β) found by the C-GRASP algorithm.

Year	β
1996	0.27
1997	0.25
1998	0.28
1999	0.25
2000	0.28
2001	0.34

Mammadov et al. [81, 82, 83] as well as our approach. Using three different metrics, we have seen that the C-GRASP approach produced a better solution than the Mammadov algorithm. One area of future research concerns discounting the non-suspected drugs of each patient by a different discount factor. This would have the effect of giving less weight to those patients having the same observed reactions and conflicting suspect drug sets. At the same time, however, this would significantly increase the number of variables in the problem. Another area of future research concerns applying our C-GRASP approach to other adverse drug reaction data. The United States Food and Drug Administration [122], the World Health Organization Uppsala Monitoring Center [123], and the Canadian Adverse Drug Reaction Information System [11] all collect adverse drug reaction reports. We are currently beginning the task of collecting data from these sites.

CHAPTER 6
OBJECT RECOGNITION OF POINTS AND LINES WITH UNKNOWN CORRESPONDENCE

6.1 Introduction

In the foreword of Hartley and Zisserman [45, p. xi], Faugeras writes ‘‘making a computer see was something that leading experts in the field of artificial intelligence thought to be at the level of difficulty of a summer student's project back in the sixties. Forty years later the task is still unsolved and seems formidable.’’ One of the main reasons for this is that the biological vision and recognition process is still largely unknown and therefore hard to emulate on computers. The field of computer vision has grown out of the research directed towards ‘making a computer see.’ In the past 15 years, a number of books have been published presenting a mathematical framework for computer vision, considering the problem from the perspective of projective geometry [28, 29, 45, 78]. In this chapter, we consider one of the still unsolved problems in computer vision, namely that of object recognition with unknown correspondence.

The general object recognition problem can be stated as follows: *suppose an object is represented by a set of m_1 points and n_1 lines. Given a picture with m_2 points and n_2 lines, does the picture contain an image of the object, under a projective transformation (i.e., pinhole camera)?* When the correspondence between object points/lines and those in the picture are known, and the scenario is error-free, solutions to this problem have been derived [38, 41, 45]. However, when the correspondence is not known, and when error is present in the scenario, then this problem becomes quite difficult to solve.

There have been some attempts to solve this object recognition problem, when certain simplifying assumptions are enforced (e.g., intensity of image points are known and invariant across images [12, 45, 78], errors are not dealt with directly in the problem formulation

[14, 112, 114], the correspondence is already known, and an inlier set is sought to determine the 'best' resulting transformation [12, 45, 78], etc.).

In this chapter, we formulate the object recognition problem for points and lines as a mixed-integer nonlinear optimization problem. In particular, we derive equations for point and line correspondence directly accounting for possible errors in the image points and lines. Using a specific type of pinhole camera, we make use of the decomposition technique of Chapter 4 to first find the best projection matrix (using the enhanced version of C-GRASP) and then apply a linear assignment algorithm to determine the 3-D to 2-D point and line correspondences. This chapter is set up as follows. In Section 6.2, we derive the equations for the point and line correspondences when error is present in the scenario and provide a mixed-integer nonlinear optimization formulation for the problem. Section 6.3 discusses a specific type of pinhole camera, briefly revisits the decomposition approach of Chapter 4, and presents computational results on randomly generated scenarios. Conclusions are provided in Section 6.4.

6.2 Object Recognition Problem Formulation

In this section we begin by deriving equations that will be satisfied for 3-D to 2-D point and line correspondences when error is not present. We will then explicitly account for possible error in the image points and lines, and provide a mixed-integer nonlinear optimization formulation. However, we begin with some notation that will be used throughout this chapter.

Let $\alpha = \{\alpha_1, \ldots, \alpha_{m_1}, \alpha'_1, \ldots, \alpha'_{n_1}\}$ represent a three-dimensional object, i.e., an unordered collection of m_1 points and n_1 lines (without loss of generality, let the unprimed α's represent the points and the primed α's represent the lines). Let $\beta = \{\beta_1, \ldots, \beta_{m_2}, \beta'_1, \ldots, \beta'_{n_2}\}$ represent a two-dimensional picture, i.e., an unordered collection of m_2 points and n_2 lines (without loss of generality, let the unprimed β's represent the points and the primed β's represent the lines). For convenience, we represent the 3-D object and 2-D image in homogeneous coordinates [38, 45, 113]. Therefore, the 3-D point $\{x, y, z\}$ is represented as

the column vector $\{x, y, z, 1\}^T$ and the 2-D point $\{u, v\}$ is represented as the column vector $\{u, v, 1\}^T$. Then, an image formed by a pinhole camera (i.e., projective transformation) can be represented as a projection from four-dimensional space to three-dimensional space, i.e., as a 3×4 matrix having 9 degrees of freedom [45]. Assume that $\Omega = [\Omega_{ij}]$ is such a 3×4 matrix.

Point Equation Derivation

For an object point $\alpha_i = \{x_i, y_i, z_i, r_i\}^T$ to correspond with the point in the picture $\beta_j = \{u_j, v_j, w_j\}^T$, using Ω as the camera, it must be true that $\Omega\alpha_i = c_{ij}\beta_j$, for some $c_{ij} \neq 0$, where c_{ij} is needed to compensate for the equivalence of homogeneous coordinates across scale. If we assume that $w_j \neq 0$ and $\Omega\alpha_i$ does not have its' third coordinate equal to 0 (i.e., $\Omega_{31}x_i + \Omega_{32}y_i + \Omega_{33}z_i + \Omega_{34}r_i \neq 0$), then $\Omega\alpha_i = c_{ij}\beta_j$ is equivalent to the system of Equations 6–1-6–2. Note that these assumptions on w_j and the third coordinate of $\Omega\alpha_i$ can be enforced in a straight-forward manner [38, 41].

$$\frac{\Omega_{11}x_i + \Omega_{12}y_i + \Omega_{13}z_i + \Omega_{14}r_i}{\Omega_{31}x_i + \Omega_{32}y_i + \Omega_{33}z_i + \Omega_{34}r_i} - \frac{u}{w} = 0 \tag{6–1}$$

$$\frac{\Omega_{21}x_i + \Omega_{22}y_i + \Omega_{23}z_i + \Omega_{24}r_i}{\Omega_{31}x_i + \Omega_{32}y_i + \Omega_{33}z_i + \Omega_{34}r_i} - \frac{u}{w} = 0 \tag{6–2}$$

If $\hat{\Omega\alpha}_i$ denotes the non-homogeneous representation of $\Omega\alpha_i$ and $\hat{\beta}_j$ represents the non-homogeneous representation of β_j, then the system of Equations 6–1-6–2 is equivalent to the vector equation $t_{ij} \equiv \hat{\Omega\alpha}_i - \hat{\beta}_j = \mathbf{0}$, which is equivalent to $\|t_{ij}\|_2 = 0$. Hence, when α_i and β_j correspond through the projection Ω, the Euclidean distance between $\hat{\Omega\alpha}_i$ and $\hat{\beta}_j$ will be 0.

Now, suppose that the non-homogeneous image point $\hat{\beta}_j$ has some error, or uncertainty, associated with it. This error can be represented as a 2×2 covariance matrix C_j (N.B.: We assume that C_j is positive semi-definite). Then, even when α_i and β_j correspond through the projection Ω, the Euclidean distance between $\hat{\Omega\alpha}_i$ and $\hat{\beta}_j$ might not be 0. This is due precisely to the uncertainty involved. To account for the uncertainty of the image

point, we generalize the Euclidean distance to the Mahalanobis distance $t_{ij}^T C_j^{-1} t_{ij}$ [45]. When this uncertainty is Gaussian in nature (it is generally assumed to be), then we can consider the Gaussian probability density function as representing the likelihood of α_i and β_j corresponding through the projection Ω, as seen in Equation 6–3. Note that this is the same technique we utilized for the sensor registration problem of Chapter 4.

$$F_{ij}(\Omega) = \frac{1}{2\pi\sqrt{|C_j|}} e^{-\frac{1}{2} t_{ij}^T C_j^{-1} t_{ij}} \tag{6–3}$$

Line Equation Derivation

Let α_i' be the object line through the two points $P_{i1} = \{x_{i1}, y_{i1}, z_{i1}, r_{i1}\}^T$ and $P_{i2} = \{x_{i2}, y_{i2}, z_{i2}, r_{i2}\}^T$ (N.B.: α_i' is defined uniquely by any two distinct points incident to α_i'). Let β_j' be the image line represented as $\beta_j' = \{a_j, b_j, c_j\}^T$, where a two-dimensional homogeneous point $q = \{u, v, w\}^T$ lies on β_j' if and only if $q^T \beta = ua_j + vb_j + wc_j = 0$. Then α_i' and β_j' correspond through the projection Ω precisely when both ΩP_{i1} and ΩP_{i2} lie on β_j', i.e., when $(\Omega P_{i1})^T \beta_j' = 0$ and $(\Omega P_{i2})^T \beta_j' = 0$. Unfortunately, this approach does not generalize to the case when there is error associated with the line β_j', as these two equations are not metric oriented. Therefore, we will look at the line case from a different perspective.

Let $\hat{\Omega} P_{i1} = \{u_{i1}, v_{i1}, 1\}^T$ denote the non-homogeneous representation of ΩP_{i1} and $\hat{\Omega} P_{i2} = \{u_{i2}, v_{i2}, 1\}^T$ denote the non-homogeneous representation of ΩP_{i2} (u_{ik} and v_{ik} are defined in Equations 6–4 and 6–5 respectively, where $k \in \{1, 2\}$).

$$u_{ik} = \frac{\Omega_{11} x_{ik} + \Omega_{12} y_{ik} + \Omega_{13} z_{ik} + \Omega_{14} r_{ik}}{\Omega_{31} x_{ik} + \Omega_{32} y_{ik} + \Omega_{33} z_{ik} + \Omega_{34} r_{ik}} \tag{6–4}$$

$$v_{ik} = \frac{\Omega_{21} x_{ik} + \Omega_{22} y_{ik} + \Omega_{23} z_{ik} + \Omega_{24} r_{ik}}{\Omega_{31} x_{ik} + \Omega_{32} y_{ik} + \Omega_{33} z_{ik} + \Omega_{34} r_{ik}} \tag{6–5}$$

It can be shown without too much difficulty that the non-homogeneous point on β_j' closest to $\Omega \hat{P}_{ik}$, call it $\hat{\delta}_{ikj}$, is given by Equation 6–6. Defining $t_{ikj} = \hat{\delta}_{ikj} - \hat{\Omega} P_{ik}$, then

we again have, in the error-free case, α'_i and β'_j corresponding through the projection Ω precisely when both $t_{i1j} = \mathbf{0}$ and $t_{i2j} = \mathbf{0}$, i.e., precisely when $\|t_{i1j}\|_2 = \|t_{i2j}\|_2 = 0$. Hence, when α'_i and β'_j correspond through the projection Ω, the Euclidean distances from $\hat{\Omega P}_{i1}$ to $\hat{\delta}_{i1j}$ and from $\hat{\Omega P}_{i2}$ to $\hat{\delta}_{i2j}$ will both be 0.

$$\hat{\delta}_{ikj} = \{\frac{b_j^2 u_{ik} - a_j b_j v_{ik} - a_j c_j}{a_j^2 + b_j^2}, \frac{a_j^2 v_{ik} - a_j b_j u_{ik} - b_j c_j}{a_j^2 + b_j^2}\}^T \tag{6–6}$$

Now, suppose that each non-homogeneous point on the image line β'_j has some error, or uncertainty, associated with it. Again, represent this uncertainty by a 2×2 covariance matrix C_j. Therefore, both $\hat{\delta}_{i1j}$ and $\hat{\delta}_{i2j}$ have associated covariance matrix C_j. With this uncertainty being Gaussian in nature we can form the likelihood of α'_i and β'_j corresponding through the projection Ω, as seen in Equation 6–7.

$$\bar{F}_{ij}(\Omega) = \frac{1}{2\pi\sqrt{|C_j|}} e^{-\frac{1}{2}(t_{i1j}^T C_j^{-1} t_{i1j} + t_{i2j}^T C_j^{-1} t_{i2j})} \tag{6–7}$$

Problem Formulation

In addition to the unknown projection matrix, Ω, we also need to determine the correspondence between the 3-D and 2-D points and lines. Hence, for each possible 3-D point α_i and 2-D point β_j, we define a binary variable ψ_{ij} to denote whether α_i and β_j should be in correspondence. We similarly define binary variables $\phi_{k\ell}$ to denote whether the 3-D line α'_k and 2-D line β'_ℓ should be in correspondence. The object recognition problem formulation is then given by Problem 6–8-6–15, where W is the space of projective transformations.

6.3 Computational Study

A general pinhole camera, Ω, can be represented by $\Omega = K[R|t]$, where K is an internal camera calibration matrix (of size 3×3, with 3 degrees of freedom), R is a 3-D rotation matrix (of size 3×3, with 3 degrees of freedom), and t is a translation vector (of size 3×1, with 3 degrees of freedom) [45].

$$\min_{\Omega\psi\phi} f(\Omega,\psi,\phi) = -\sum_{i=1}^{m_1}\sum_{j=1}^{m_2}\psi_{ij}F_{ij}(\Omega) - \sum_{k=1}^{n_1}\sum_{\ell=1}^{n_2}\phi_{k\ell}\bar{F}_{k\ell}(\Omega) \tag{6–8}$$

$$\ni \sum_{i=1}^{m_1}\psi_{ij} \leq 1 \;\forall\; j \in \{1,\ldots,m_2\} \tag{6–9}$$

$$\sum_{j=1}^{m_2}\psi_{ij} \leq 1 \;\forall\; i \in \{1,\ldots,m_1\} \tag{6–10}$$

$$\sum_{k=1}^{n_1}\phi_{k\ell} \leq 1 \;\forall\; \ell \in \{1,\ldots,n_2\} \tag{6–11}$$

$$\sum_{\ell=1}^{n_2}\phi_{k\ell} \leq 1 \;\forall\; k \in \{1,\ldots,n_1\} \tag{6–12}$$

$$\psi_{ij} \in \{0,1\} \;\forall\; i \in \{1,\ldots,m_1\}, \forall\; j \in \{1,\ldots,m_2\} \tag{6–13}$$

$$\phi_{k\ell} \in \{0,1\} \;\forall\; k \in \{1,\ldots,n_1\}, \forall\; \ell \in \{1,\ldots,n_2\} \tag{6–14}$$

$$\Omega \in W \tag{6–15}$$

When K is set to the identity matrix, then the resulting projection matrix Ω preserves distances between points, precisely what is needed to utilize the decomposition approach of Chapter 4. This decomposition approach solves Problem 6–8-6–15 by first solving Problem 6–16 for Ω, using enhanced C-GRASP, and then applying a linear assignment algorithm to determine the binary assignment variables ψ_{ij} and $\phi_{k\ell}$.

$$\min_{\Omega\in W} \mathbf{F}(\Omega) = -\sum_{i=1}^{m_1}\sum_{j=1}^{m_2}F_{ij}(\Omega) - \sum_{k=1}^{n_1}\sum_{\ell=1}^{n_2}\bar{F}_{k\ell}(\Omega) \tag{6–16}$$

To illustrate our approach for solving the object recognition optimization problem 6–8-6–15, we consider three scenarios classes. Each class is designated by the number of 3-D points, the number of 2-D points, the number of 3-D lines, the number of 2-D lines, the number of truthful point correspondences, the number of truthful line correspondences, and the maximum error (in pixels) associated with the 2-D data. The details for each scenario class are given in Table 6–1.

Table 6–1: Object recognition scenario classes.

Scenario	Points				Lines			
Class	3D	2D	True Corr.	Max Error	3D	2D	True Corr.	Max Error
1	15	20	11	0.5	-	-	-	-
2	12	16	8	0.5	8	11	5	0.5
3	-	-	-	-	16	25	13	0.5

Table 6–2: Object recognition C-GRASP parameters.

Number Multi-Starts	100
h_s	0.10
h_e	0.01
ρ_{lo}	0.70

For each class, 100 randomly generated scenarios were created. Each randomly generated scenario was input to the enhanced C-GRASP algorithm, and run for a fixed number of multi-starts. The C-GRASP parameters are listed in Table 6–2. When the C-GRASP algorithm completed the run for a scenario, using the best projection matrix found, Ω, the 3-D points and lines from the scenario were projected onto the 2-D image plane, and the JVC linear assignment algorithm [65] was applied to determine the optimal assignment of 3-D points and lines to those in 2-D.

For each scenario class, we averaged (over all the scenarios in the class) the time needed (in minutes) for the C-GRASP algorithm to complete all multi-starts (ATT), the time (in minutes) when C-GRASP found the best Ω (ABT), the multi-start iteration when C-GRASP found the best Ω (ABI), the number of assignments of 3-D points and lines to 2-D points and lines that were correct (ANCA), and the distances between the projected 3-D points and lines and their truthful corresponding 2-D points and lines (ADC). These results are shown in Table 6–3. As is clear from the table, for all three scenario classes, the decomposition approach performs very well. Of note is that adding lines to the scenario seems to add significant complexity to the surface of the objective function, thus taking longer for the C-GRASP algorithm to run to completion and longer on average to locate the correct solution (from a time and multi-start perspective).

Table 6–3: Object recognition test results.

Scenario				Points		Lines	
Class	ATT	ABT	ABI	ANCC	ADC	ANCA	ADC
1	9.11	4.97	54.7	9.1	1.12	-	-
2	13.17	8.31	63.6	7.2	0.81	4.5	0.05
3	30.58	18.28	57.6	-	-	8.4	0.76

6.4 Conclusions

In this chapter, we have examined a problem from computer vision: the recognition of a 3-D object, represented by points and lines, in an image, when the correspondence of points and lines is not known *a priori*. We have formulated this problem as a mixed-integer nonlinear optimization problem, explicitly accounting for possible noise in the 2-D points and lines. For a pinhole camera (with specific internal calibration parameters), we decomposed the mixed-integer optimization problem into first determining the best projection matrix transforming the 3-D points and lines onto the 2D picture and then using a linear assignment algorithm to determine the correspondences between the points and lines of the object and those of the image. Computational studies have shown that this approach does a very good job of determining the correct projection matrix and correspondences. Future research will be geared towards reducing the time of the C-GRASP algorithm, to make this approach suitable for a real-time system.

CHAPTER 7
CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

In this work, we have presented a new continuous global optimization heuristic, C-GRASP. Chapter 2 detailed two versions of this heuristic and showed C-GRASP to be superior to three other heuristics on a set of standard benchmark functions. In addition, when implemented with sequential stopping rules, C-GRASP performed extremely well on all test functions. C-GRASP was used to find all roots to systems of nonlinear equations in Chapter 3. For all systems considered, C-GRASP almost always found all of the roots in the feasible region. We applied the enhanced version of C-GRASP to a sensor registration problem in Chapter 4. Our approach was shown to be better than two others that have been integrated into fielded military systems. Adverse drug reactions, a very important problem in quality health-care, was considered in Chapter 5. We showed that C-GRASP out-performed an approach by Mammadov et al. [81, 82, 83] using three different metrics. In Chapter 6, we applied C-GRASP to an open problem in computer vision which produced promising results.

The problems examined in this research represent important areas of the applied sciences. However, these problems only represent a small subset of the challenging problems that are in need of solutions. Here I discuss some immediate extensions of this work:

- In its current form, the C-GRASP construction procedure samples the objective function along the directions defined by the coordinate axes. When the objective function is completely or partially separable, sampling along the coordinate axes appears to be very effective. However, when the objective function is not separable (e.g., the Zakharov function), then this approach might result in slow convergence to the global optimum.
  Research has already been started to investigate modifications to the construction procedure where, instead of sampling along the coordinate axis directions, an orthogonal set of directions is randomly chosen each time the construction procedure

is called (for an n-dimensional problem, a random $n \times n$ matrix can be transformed into a basis for $\mathbb{R}^n$, except on a set of measure 0) [39, 61].

- The C-GRASP algorithm is currently only set up to deal with optimization problems containing box constraints. However, many optimization problems have more complicated constraints than those of the box type. Solution techniques to these optimization problems generally fall into three categories: i) adding the constraints to the objective function as penalty terms, ii) lagrangian relaxation, and iii) multi-objective optimization. Research will be undertaken to implement versions of C-GRASP for multi-objective optimization and Lagrangian relaxation.

- In the recent literature, many of the heuristics considered have in fact been hybrid heuristics, i.e., combining more than one traditional heuristic to produce an algorithm that is superior to those traditional heuristics [48, 49, 120, 130].
  Two hybrid C-GRASP heuristics are currently being researched. The first combines C-GRASP with a simulated annealing local search algorithm. In this algorithm, the construction procedure would proceed as in the current C-GRASP algorithm. However, the local improvement procedure would operate in a simulated annealing manner. When the candidate solution x' is formed, if $f(x') < f(x^*)$, then as in the current local improvement algorithm, we update x^* with x'. However, if $f(x') \geq f(x^*)$, then with a certain probability, we still update x^* with x'.
  The second hybrid approach considered is to combine C-GRASP with a genetic algorithm. In this implementation, all members of the population independently perform an iteration of C-GRASP. When all individuals have returned from local improvement, genetic operators are applied to form the next generation of the population.

- When real-world problems are formulated mathematically, in many instances the formulated optimization problem will include discrete, as well as continuous, variables. We encountered two such problems (seen in Chapters 4 and 6). For these problems, we were able to apply a decomposition approach which allowed us to first deal with the continuous variables, and then, using the resulting values of the continuous variables, solve for the discrete variables. Unfortunately, for many mixed-integer optimization problems, a decomposition approach is not theoretically available. Hence, we must deal with both the continuous and discrete variables at the same time. It is our intent to focus some research effort into combining the C-GRASP algorithm with a discrete version of GRASP in order to develop a suitable heuristic for mixed-integer optimization problems.

APPENDIX
CONTINUOUS GLOBAL OPTIMIZATION TEST FUNCTIONS

This appendix contains the definition of all test functions used in Chapter 2.

(A_n) **Ackley Function**

Definition: $A_n(x) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)} + 20 + e$

Domain: $[-15, 30]^n$

Global Minimum: $x^* = (0, \ldots, 0)$; $A_n(x^*) = 0$

(BE) **Beale Function**

Definition: $BE(x) = (1.5 - x_1 - x_1x_2)^2 + (2.25 - x_1 - x_1x_2^2)^2 + (2.625 - x_1 - x_1x_2^3)^2$

Domain: $[-4.5, 4.5]^2$

Global Minimum: $x^* = (3, \frac{1}{2})$; $BE(x^*) = 0$

(B_2) **Bohachevsky Function**

Definition: $B_2(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$

Domain: $[-50, 100]^2$

Global Minimum: $x^* = (0, 0)$; $B_2(x^*) = 0$

(BO) **Booth Function**

Definition: $BO(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$

Domain: $[-10, 10]^2$

Global Minimum: $x^* = (1, 3)$; $BO(x^*) = 0$

(*BR*) **Branin Function**

Definition: $BR(x) = (x_2 - \frac{5}{4\pi^2}x_1^2 + \frac{1}{\pi}5x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$

Domain: $[-5, 15]^2$

Global Minimum (one of several): $x^* = (\pi, 2.275)$; $BR(x^*) = 0.397887$

(*CV*) **Colville Function** (also called **Wood Function**)

Definition: $CV(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x_3)^2$
$+10.1\left[(x_2 - 1)^2 + (x_4 - 1)^2\right] + 19.8(x_2 - 1)(x_4 - 1)$

Domain: $[-10, 10]^4$

Global Minimum: $x^* = (1, 1, 1, 1)$; $CV(x^*) = 0$

(DP_n) **Dixon and Price Function**

Definition: $DP_n(x) = (x_1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$

Domain: $[-10, 10]^n$

Global Minimum: $x_i^* = 2^{-\frac{2^i - 2}{2^i}}$ $\forall i = 1, \ldots, n$; $DP_n(x^*) = 0$

(*EA*) **Easom Function**

Definition: $EA(x) = -\cos(x_1)\cos(x_2)e^{-(x_1 - \pi)^2 - (x_2 - \pi)^2}$

Domain: $[-100, 100]^2$

Global Minimum: $x^* = (\pi, \pi)$; $EA(x^*) = -1$

(*GP*) **Goldstein and Price Function**

Definition: $GP(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)]$
$[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$

Domain: $[-2, 2]^2$

Global Minimum: $x^* = (0, -1)$; $GP(x^*) = 3$

(GR_n) **Griewank Function**

Definition: $GR_n(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos(\frac{x_i}{\sqrt{i}}) + 1$

Domain: $[-300, 600]^n$

Global Minimum: $x^* = (0, \ldots, 0)$; $GR_n(x^*) = 0$

($H_{n,m}$) **Hartmann Function**

Definition: $H_{n,m}(x) = -\sum_{i=1}^{m} \alpha_i e^{-\sum_{j=1}^{n} A_{ij}^{(n)} (x_j - P_{ij}^{(n)})^2}$

Domain: $[0, 1]^n$

Global Minimum: $(n = 3,\ m = 4)\ x^* = (0.114614, 0.555649, 0.852547)$; $H_{3,4}(x^*) = -3.86278$

Global Minimum: $(n = 6,\ m = 4)\ x^* = (0.201690, 0.150011, 0.476874, 0.275332, 0.311652, 0.657300)$;

$H_{6,4}(x^*) = -3.32237$

Parameters:

$$A^{(3)} = \begin{bmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}$$

$$P^{(3)} = 10^{-4} \begin{bmatrix} 6890 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8838 \end{bmatrix}$$

$$A^{(6)} = \begin{bmatrix} 10 & 3 & 17 & 3.05 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$$

$$P^{(6)} = 10^{-4} \begin{bmatrix} 1312 & 1696 & 5569 & 124 & 8283 & 5886 \\ 2329 & 4135 & 8307 & 3736 & 1004 & 9991 \\ 2348 & 1451 & 3522 & 2883 & 3047 & 6650 \\ 4047 & 8828 & 8732 & 5743 & 1091 & 381 \end{bmatrix}$$

$\alpha = [1, 1.2, 3, 3.2]$

(L_n) Levy Function

Definition: $L_n(x) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1} \left[(y_i - 1)^2 (1 + 10\sin^2(\pi y_i + 1)) \right] + (y_n - 1)^2 (1 + 10\sin^2(2\pi y_n))$

Domain: $[-10, 10]^n$

Global Minimum: $x^* = (1, \ldots, 1)$; $L_n(x^*) = 0$

Parameters: $y_i = 1 + \frac{x_i - 1}{4} \; \forall \; i = 1, \ldots, n$

(M) Matyas Function

Definition: $M(x) = 0.26(x_1^2 + x_2^2) - 0.48 x_1 x_2$

Domain: $[-5, 10]^2$

Global Minimum: $x^* = (0, 0)$; $M(x^*) = 0$

($P_{n,\beta}$) Perm Function

Definition: $P_{n,\beta}(x) = \sum_{k=1}^{n} \left[\sum_{i=1}^{n} (i^k + \beta)((\frac{x_i}{i})^k - 1) \right]^2$

Domain: $[-n, n]^n$

Global Minimum: $x^* = (1, 2, \ldots, n)$; $P_{n,\beta}(x^*) = 0$

($P^0_{n,\beta}$) Perm$_0$ Function

Definition: $P^0_{n,\beta}(x) = \sum_{k=1}^{n} [\sum_{i=1}^{n} (i + \beta)((x_i^k - (\frac{1}{i})^k))]^2$

Domain: $[-n, n]^n$

Global Minimum: $x^* = (1, \frac{1}{2}, \ldots, \frac{1}{n})$; $P^0_{n,\beta}(x^*) = 0$

(PW_n) **Powell Function**

Definition: $PW_n(x) = \sum_{i=1}^{\frac{n}{4}} \Big[(x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4 \Big]$

Domain: $[-4,5]^n$

Global Minimum: $x^* = (3,-1,0,1,3,\ldots,3,-1,0,1)$; $PW_n(x^*) = 0$

($PS_{n,b}$) **Power Sum Function**

Definition: $PS_{n,b}(x) = \sum_{k=1}^{n}((\sum_{i=1}^{n} x_i^k) - b_k)^2$

Domain: $[0,n]^n$

Global Minimum ($n = 4, b = \{8,18,44,114\}$): $x^* = (1,2,2,3)$; $PS_{4,\{8,18,44,114\}}(x^*) = 0$

(RA_n) **Rastrigin Function**

Definition: $RA_n(x) = 10n + \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i))$

Domain: $[-2.56,5.12]^n$

Global Minimum: $x^* = (0,\ldots,0)$; $RA_n(x^*) = 0$

(R_n) **Rosenbrock Function**

Definition: $R_n(x) = \sum_{j=1}^{n-1}\left[100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2\right]$

Domain: $[-10,10]^n$

Global Minimum: $x^* = (1,\ldots,1)$; $R_n(x^*) = 0$

(SC_n) **Schwefel Function**

Definition: $SC_n(x) = 418.9829n - \sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|})$

Domain: $[-500,500]^n$

Global Minimum: $x^* = (420.9687,\ldots,420.9687)$; $SC_n(x^*) = 0$

($S_{4,m}$) **Shekel Function**

Function Definition: $S_{4,m}(x) = -\sum_{i=1}^{m}[(x-a_i)^T(x-a_i)+c_i]^{-1}$

Domain: $[0,10]^4$

Global Minimum: $x^* = (4,4,4,4)$; $S_{4,5}(x^*) = -10.15319538$, $S_{4,7}(x^*) = -10.40281868$, and $S_{4,10}(x^*) = -10.53628349$

Parameters:

$$a = \begin{bmatrix} 4.0 & 4.0 & 4.0 & 4.0 \\ 1.0 & 1.0 & 1.0 & 1.0 \\ 8.0 & 8.0 & 8.0 & 8.0 \\ 6.0 & 6.0 & 6.0 & 6.0 \\ 7.0 & 3.0 & 7.0 & 3.0 \\ 2.0 & 9.0 & 2.0 & 9.0 \\ 5.0 & 5.0 & 3.0 & 3.0 \\ 8.0 & 1.0 & 8.0 & 1.0 \\ 6.0 & 2.0 & 6.0 & 2.0 \\ 7.0 & 2.6 & 7.0 & 3.6 \end{bmatrix}$$

$c = [0.1, 0.2, 0.2, 0.4, 0.4, 0.6, 0.3, 0.7, 0.5, 0.5]$

(SH) **Shubert Function**

Definition: $SH(x) = \left[\sum_{i=1}^{5} i\cos[(i+1)x_1+i]\right]\left[\sum_{i=1}^{5} i\cos[(i+1)x_2+i]\right]$

Domain: $[-10,10]^2$

Global Minimum (one of several): $x^* = (5.48242188, 4.85742188)$; $SH(x^*) = -186.7309$

(CA) **Six-Hump CamelBack Function**

Definition: $CA(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$

Domain: $[-5,5]^2$

Global Minimum (one of several): $x^* = (0.08984375, -0.71289062)$; $CA(x^*) = -1.03162801$

(SP_n) **Sphere Function**

Definition: $SP_n(x) = \sum_{i=1}^{n} x_i^2$

Domain: $[-2.56, 5.12]^n$

Global Minimum: $x^* = (0, \ldots, 0)$; $SP_n(x^*) = 0$

N.B.: The De Joung Function (DJ) is just a special case of the Sphere Function, i.e $DJ(x) = SP_3(x)$

(SS_n) **Sum of Squares Function**

Definition: $SS_n(x) = \sum_{i=1}^{n} ix_i^2$

Domain: $[-5, 10]^n$

Global Minimum: $x^* = (0, \ldots, 0)$; $f(x^*) = 0$

(T_n) **Trid Function**

Definition: $T_n(x) = \sum_{i=1}^{n} (x_i - 1)^2 - \sum_{i=2}^{n} x_i x_{i-1}$

Domain: $[-n^2, n^2]^n$

Global Minimum: $x_i^* = i(n + 1 - i) \; \forall \; i = 1, \ldots, n$; $T_6(x^*) = -50$ and $T_{10}(x^*) = -210$

(Z_n) **Zakharov Function**

Definition: $Z_n(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4$

Domain: $[-5, 10]^n$

Global Minimum: $x^* = (0, \ldots, 0)$; $Z_n(x^*) = 0$

REFERENCES

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

[2] J. S. Almenoff, W. DuMouchel, L. A. Kindman, X. Yang, and D. Fram. Disporportionality analysis using empirical Bayes data mining: a tool for the evaluation of drug interactions in the post-marketing setting. *Pharmacoepidemiology and Drug Safety*, 12:517–521, 2003.

[3] Australian Department of Health and Ageing Therapeutic Goods Administration. Australian Adverse Drug Reactions Bulliten. Last time accessed: August 2006. http://www.tga.gov.au/adr/aadrb.htm.

[4] J. Barhen, V. Protopopescu, and D. Reister. TRUST: A deterministic algorithm for global optimization. *Science*, 276:1094–1097, 1997.

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

[6] 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.

[7] S. Blackman and R. Popoli. *Design and Analysis of Modern Tracking Systems*. Artech House, Norwood, MA, 1999.

[8] C. G. E. Boender and H. E. Romeijn. Stochastic methods. In R. Horst and P. M. Pardalos, editors, *Handbook of Global Optimization*, pages 829–869. Kluwer Academic Publishers, 1995.

[9] I. O. Bohachevsky, M. E. Johnson, and M. L. Stein. generalized simulated annealing for function optimization. *Technometrics*, 28(3):209–217, 1986.

[10] R. A. R. Butler and E. E. Slaminka. An evaluation of the sniffer global optimization algorithm using standard test functions. *Journal of Computational Physics*, 99(1):28–32, 1992.

[11] Canadian Adverse Drug Reaction Information System. Canadian Adverse Drug Reaction Database. Last time accessed: August 2006. http://www.cbc.ca/news/adr/database/.

[12] J. Chai and S. D. Ma. Robust epipolar geometry estimation using genetic algorithm. *Pattern Recognition Letters*, 19:829–838, 1998.

[13] L. Chen, T. Zhou, and Y. Tang. Protein structure alignment by deterministic annealing. *Bioinformatics*, 21(1):51–62, 2005.

[14] Y. Cheng, V. Wu, R. T. Collins, A. R. Hanson, and E. M. Riseman. Maximum weight bipartite matching technique and its application to solve image feature matching. In *Proc. of the SPIE Conference on Visual Communication and Image Processing*, 1996.

[15] T. Chung, L. Cremean, W. B. Dunbar, Z. Jin, E. Klavins, D. Moore, A. Tiwari, D. Van Gogh, and S. Waydo. A platform for cooperative and coordinated control of multiple vehicles: The Caltech multi-vehivle wireless testbed. In S. Butenko, R. Murphey, and P. M. Pardalos, editors, *Cooperative Control and Optimization*, pages 75–104. Kluwer Academic Publishers, 2004.

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

[17] A. Corana, M. Marchesi, C. Martini, and S. Ridella. Minimizing multimodal functions of continuous variables with the simulated annealing algorithm. *ACM Transactions on Mathematical Software*, 13(3):262–280, 1987.

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

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

[20] D. Cvijović and J. Klinowski. Taboo search: An approach to the multiple minima problem. *Science*, 267:664–666, 1995.

[21] D. Cvijović and J. Klinowski. Taboo search: An approach to the multiple minima problem for continuous functions. In P. M. Pardalos and H. E. Romeijn, editors, *Handbook of Global Optimization, volume 2*, pages 387–406. Kluwer Academic Publishers, 2002.

[22] M. Dana. Registration: A prerequisite for multiple sensor tracking. In Y. Bar-Shalom, editor, *Multitarget-Multisensor Tracking: Advanced Applications*, pages 155–185. Artech House, Norwood, MA, 1990.

[23] J. G. Digalakis and K. G. Margaritis. An experimental study of benchmarking functions for genetic algorithms. *International Journal of Computer Mathematics*, 79(4):403–416, 2002.

[24] L. C. W. Dixon and G. P. Szego. The global optimization problem: an introduction. In L. C. W. Dixon and G. P. Szego, editors, *Towards Global optimization 2*, pages 1–15. North-Holland, Amsterdam, 1978.

[25] C. C. Y. Dorea. Stopping rules for a random optimization method. *SIAM Journal on Control and Optimization*, 28(4):841–850, 1990.

[26] J. P. K. Doyle and D. J. Wales. Thermodynamics of global optimization. *Physical Review Letters*, 80(7):1357–1360, 1998.

[27] Y. G. Evtushenko and M. A. Potapov. Deterministic global optimization. In E. Spedicato, editor, *Continuous Optimization: The State or Art*, pages 481–500. Kluwer, 1994.

[28] O. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, 1993.

[29] O. Faugeras and Q. T. Luong. *The Geometry of Multiple Images: The laws that govern the formation of multiple images of a scene and some of their applications*. The MIT Press, 2001.

[30] T. A. Feo and M. G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.

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

[32] 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.

[33] 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.

[34] C. A. Floudas and P. M. Pardalos. A collection of test problems for constrained global optimization algorithms. In G. Goods and J. Hartmanis, editors, *Lecture Notes in Computer Science*, volume 455. Springer Verlag, Berlin, 1990.

[35] 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.

[36] F. Freudenstein. Kinematics: Past, present and future. *Mechanism and Machine Theory*, 8(2):151–161, 1973.

[37] A. Gelb. *Applied Optimal Estimation*. MIT Press, 1974.

[38] R. F. Gleeson, F. D. Grosshans, M. J. Hirsch, and R. M. Williams. Algorithms for the recognition of 2D images of *m* points and *n* lines in 3D. *Image and Vision Computing*, 21:497–504, 2003.

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

[40] L. Granvilliers and F. Benhamou. Progress in the solving of a circuit design problem. *Journal of Global Optimization*, 20(2):155–168, 2001.

[41] F. D. Grosshans. On the equations relating a three-dimensional object and its two-dimensional images. *Advances in Applied Mathematics*, 34:366–392, 2005.

[42] U. H. E. Hansmann and L. T. Wille. Global optimization by energy landscape paving. *Physical Review Letters*, 88(6):068105, 2002.

[43] W. E. Hart. Adaptive global optimization with local search. *PhD. thesis, University of California, San Diego*, 1994.

[44] W. E. Hart. Sequential stopping rules for random optimization methods with application to multistart local search. *SIAM Journal of Optimization*, 9(1):270–290, 1998.

[45] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[46] J. T. Harvey, C. Turville, and S. M. Barty. Data mining of the Australian adverse drug reactions database: a comparison of Bayesian and other statistical indicators. *International Transactions in Operations Research*, 11:419–433, 2004.

[47] D. W. Hearn and M. V. Ramana. Solving congestion toll pricing models. In P. Marcotte and S. Nguyen, editors, *Equilibrium and Advanced Transportation Modeling*, pages 109–124. Kluwer Academic Publishers, 1998.

[48] A. R. Hedar and M. Fukushima. Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization. *Optimization Methods and Software*, 17:891–912, 2002.

[49] A. R. Hedar and M. Fukushima. Minimizing multimodal functions by simplex coding genetic algorithms. *Optimization Methods and Software*, 18:265–282, 2003.

[50] A. R. Hedar and M. Fukushima. Tabu search directed by direct search methods for nonlinear global optimization. *European Journal of Operational Research*, 170:329–349, 2006.

[51] R. E. Helmick and T. R. Rice. Alignment of a 2-D sensor and a 3-D radar. In *Acquisition, Tracking, and Pointing VI (Proc. SPIE)*, volume 1697, pages 142–157, 1992.

[52] M. J. Hirsch. A novel, generalized gridlock correction algorithm. In *Proc. National Fire Control Symposium*, Lihue, HI, 2004.

[53] M. J. Hirsch, W. Barker, B. Blyth, Y. Chen, R. P. DeLong, and C. Eck. Advanced techniques in track identification and correlation. In *Proc. National Fire Control Symposium*, Lihue, HI, 2004.

[54] M. J. Hirsch, C. N. Meneses, P. M. Pardalos, M. Ragle, and M. G. C. Resende. A Continuous GRASP to determine the relationship between drugs and adverse reactions. *Submitted to Annals of Biomedical Engineering*, 2006.

[55] 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.

[56] M. J. Hirsch, H. Ortiz-Pena, and R. Cooperman. Smoothing techniques using an IMM and multi-sensor tracking with time-late data. In *Proc. National Symposium on Sensor and Data Fusion*, Laurel, MD, 2004.

[57] M. J. Hirsch, P. M. Pardalos, and M. G. C. Resende. 2D point/line recognition of projected 3D objects with unknown correspondence using a continuous GRASP. *Submitted to International Journal of Computer Vision*, 2006.

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

[59] M. J. Hirsch, P. M. Pardalos, and M. G. C. Resende. Solving systems of nonlinear equations using Continuous GRASP. *Submitted to Journal of Heuristics*, 2006.

[60] M. J. Hirsch, P. M. Pardalos, and M. G. C. Resende. Speeding up Continuous GRASP. *Submitted to European Journal of Operational Research*, 2006.

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

[62] R. Horst, P. M. Pardalos, and N. V. Thoai. *Introduction to Global Optimization*. Kluwer Academic Publishers, Dordrecht, 2nd edition, 2000.

[63] R. Horst and H. Tuy. *Global Optimization: Deterministic Approaches*. Springer-Verlag, Heidelberg, 2nd edition, 1993.

[64] Y. Ji, H. Ying, J. Yen, S. Zhu, R. M. Massanari, and D. C. Barth-Jones. Team-based multi-agent system for early detection of adverse drug reactions in postmarketing surveilance. In *Annual Meeting of the North American Fuzzy Information Processing Society*, pages 644–649, 2005.

[65] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 39:325–340, 1987.

[66] S. Kazamias, D. Douillet, F. Weihe, C. Valentin, A. Rousse, S. Sebban, G. Grillon, F. Augé, D. Hulin, and P. Balcou. Global optimization of high harmonic generation. *Physical Review Letters*, 90(19):193901, 2003.

[67] R. B. Kearfott. Some tests of generalized bisection. *ACM Transactions on Mathematical Software*, 13(3):197–220, 1987.

[68] R. Kenefic. Local and remote track file registration using minimum description length. *IEEE Transactions on Aerospace and Electronic Systems*, 29(3):651–655, 1993.

[69] G. H. Koon and A. V. Sebald. Some interesting test functions for evaluating evolutionary programming strategies. In *Proc. of the Fourth Annual Conference on Evolutionary Programming*, pages 479–499, 1995.

[70] P. Krokhmal, R. Murphey, P. M. Pardalos, and S. Uryasev. Use of conditional value-at-risk in stochastic programs with poorly defined distributions. In S. Butenko, R. Murphey, and P. M. Pardalos, editors, *Recent Developments in Cooperative Control and Optimization*, pages 225–243. Kluwer Academic Publishers, 2004.

[71] P. Krokhmal, R. Murphey, P. M. Pardalos, S. Uryasev, and G. Zrazhevsky. Robust decision making: Addressing uncertainties in distributions. In S. Butenko, R. Murphey, and P. M. Pardalos, editors, *Cooperative Control: Models, Applications, and Algorithms*, pages 165–185. Kluwer Academic Publishers, 2003.

[72] 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.

[73] M. Laguna and R. Martí. Experimental testing of advanced Scatter Search designs for global optimization of multimodal functions. *Journal of Global Optimization*, 33:235–255, 2005.

[74] J. Lee, I. H. Lee, and J. Lee. Unbiased global optimization of Lennard-Jones clusters for $N \leq 201$ using the conformational space annealing method. *Physical Review Letters*, 91(8):080201, 2003.

[75] 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.

[76] M. Locatelli. Simulated annealing algorithms for continuous global optimization. In P. M. Pardalos and H. E. Romeijn, editors, *Handbook of Global Optimization, volume 2*, pages 179–229. Kluwer Academic Publishers, 2002.

[77] D. G. Luenberger. *Investment Science*. Oxford University Press, 1998.

[78] Y. Ma, S. Soatto, J. Kosecka, and S. S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer-Verlag, 2004.

[79] M. Mammadov and A. Banerjee. An optimization approach identifying drugs responsible for adverse drug reactions. In J. Ryan, P. Manyem, K. Sugeng, and M. Miller, editors, *Proc. of the Sixteenth Australasian Workshop on Combinatorial Algorithms*, pages 185–200, 2005.

[80] M. Mammadov and A. Banerjee. An optimization approach to the study of drug-drug interactions. In J. Ryan, P. Manyem, K. Sugeng, and M. Miller, editors, *Proc. of the Sixteenth Australasian Workshop on Combinatorial Algorithms*, pages 201–216, 2005.

[81] M. Mammadov, E. Dekker, and G. Saunders. An optimization approach to the study of drug-reaction relationships on the basis of adrac dataset: Neurological class of reactions. In A. Rubinov and M. Sniedovich, editors, *Proc. of The Sixth International Conference on Optimization: Techniques and Applications (ICOTA6)*, 2004.

[82] M. Mammadov, A. Rubinov, and J. Yearwood. An optimization approach to identify the relationship between features and output of a multi-label classifier. In A. Rubinov and M. Sniedovich, editors, *Proc. of The Sixth International Conference on Optimization: Techniques and Applications (ICOTA6)*, 2004.

[83] M. Mammadov and G. Saunders. A comparison of two methods to establish drug-reaction relationships in the adrac database. In S. S. Raza Abidi, editor, *Proc. of The Fourth International ICSC Symposium on ENGINEERING OF INTELLEGENT SYSTEMS (EIS 2004)*, 2004.

[84] D. Manocha and J. F. Canny. Efficient inverse kinematics for general 6R manipulators. *IEEE Transactions on Robotics and Automation*, 10(5):648–657, 1994.

[85] 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.

[86] 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.

[87] D. Maurer. Efficient Radar-to-IR bias estimation and correlation. In *Proc. 11th AIAA/MDA Technology Conference*, 2002.

[88] J. R. McDonnell and D. E. Waagen. An empirical study of recombination in evolutionary search. In *Proc. of the Fourth Annual Conference on Evolutionary Programming*, pages 465–478, 1995.

[89] K. Meintjes and A. P. Morgan. A methodology for solving chemical equilibrium systems. *Applied Mathematics and Computation*, 22:333–361, 1987.

[90] K. Meintjes and A. P. Morgan. Element variables and the solution of complex chemical equilibrium problems. *Combustion Science and Technology*, 68:35–48, 1989.

[91] K. Meintjes and A. P. Morgan. Chemical equilibrium systems as numerical test problems. *ACM Transactions on Mathematical Software*, 16(2):143–151, 1990.

[92] J. P. Merlet. The COPRIN examples page. Last time accessed: August 2006. http://www-sop.inria.fr/coprin/logiciels/ALIAS/Benches/benches.html.

[93] Z. Michalewicz and D. B. Fogel. *How to Solve It: Modern Heuristics*. Springer Verlag, Berlin, 2nd edition, 2004.

[94] Z. Michalewicz and G. Nazhiyath. Genocop III: A Co-evolutionary Algorithm for Numerical Optimization Problems with Nonlinear Constraints. In *Proc. of the 2nd IEEE ICEC*, Perth, Australia, 1995.

[95] M. D. Miller and O. E. Drummond. Coordinate transformation bias in target tracking. In O. E. Drummond, editor, *Signal and Data Processing of Small Targets (Proc. SPIE)*, volume 3809, pages 409–424, 1999.

[96] N. Mladenović, J. Petrović, V. Kovacević-Vujcić, and M. Cangalović. Solving spread spectrum radar polyphase code design problem by tabu search and variable neighborhood search. *European Journal of Operational Research*, 151(2):389–399, 2003.

[97] 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.

[98] J. J. Moré and M. Y. Cosnard. Numerical solution of nonlinear equations. *ACM Transactions on Mathematical Software*, 5(1):64–85, 1979.

[99] J. J. Moré, B. S. Garbow, and K. E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 7(1):17–41, 1981.

[100] A. P. Morgan, A. J. Sommese, and C. W. Wampler. Polynomial continuation for mechanism design problems. In E. L. Allgower and K. Georg, editors, *Computational Solution of Nonlinear Systems of Equations*, pages 495–517. American Mathematical Society, 1990.

[101] A. Neumaier. The COCONUT Benchmark. Last time accessed: April 2006. http://www.mat.univie.ac.at/~neum/glopt/coconut/Benchmark/Benchmark.html.

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

[103] R. V. Pappu, R. K. Hart, and J. W. Ponder. Analysis and application of potential energy smoothing and search methods for global optimization. *Journal of Physical Chemistry* ***B***, 102:9725–9742, 1998.

[104] P. M. Pardalos and M. G. C. Resende, editors. *Handbook of Applied Optimization*. Oxford University Press, 2002.

[105] D. T. Pham and D. Karaboga. *Intelligent Optimization Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing, and Neural Networks*. Springer-Verlag, London, 2000.

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

[107] H. Ratschek and J. Rokne. Experiments using interval analysis for solving a circuit design problem. *Journal of global optimization*, 3(3):501–518, 1993.

[108] M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 219–249. Kluwer Academic Publishers, 2003.

[109] A. H. G. Rinnooy Kan and G. T. Timmer. Stochastic methods for global optimization. *American Journal of Mathematical and Management Sciences*, 4:7–40, 1984.

[110] E. Roux, F. Thiessard, A. Fourrier, B. Begaud, and P. Tubert-Bitter. Evaluation of statistical association measures for the automatic signal generation in pharmacovigilance. *IEEE Transactions on Information Technology in Biomedicine*, 9(4):419–433, 2005.

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

[112] G. L. Scott and H. C. Longuet-Higgins. An algorithm for associating the features of two images. *Proc. of the Royal Society of London* ***B***, 244:21–26, 1991.

[113] J. G. Semple and G. T. Kneebone. *Algebraic Projective Geometry*. Oxford University Press, 1952.

[114] L. S. Shapiro and J. M. Brady. Feature based correspondence: an eigenvector approach. *Image and Vision Computing*, 10(5):283–288, 1992.

[115] P. Siarry, G. Berthiau, F. Durbin, and J. Haussy. Enhanced simulated annealing for globally minimizing functions of many continuous variables. *ACM Transactions on Mathematical Software*, 23(2):209–228, 1997.

[116] J. E. Smith. Genetic algorithms. In P. M. Pardalos and H. E. Romeijn, editors, *Handbook of Global Optimization, volume 2*, pages 275–362. Kluwer Academic Publishers, 2002.

[117] D. G. Sotiropoulos, V. P. Plagianakos, and M. N. Vrahatis. An evolutionary algorithm for minimizing multimodal functions. In E. A. Lipitakis, editor, *Proc. of the 5th Hellenic-European Conference on Computer Mathematics and Its Applications (HERCMA2001)*, volume 2, pages 496–500, 2002.

[118] J. J. Sudano. A least square algorithm with covariance weighting for computing the translational and rotational errors between two radar sites. In *Proc. of the IEEE*

Aerospace and Electronics Conference (NAECOM 93), volume 1, pages 383–387, 1993.

[119] A. Torn and A. Zilinskas. Global optimization. In *Lecture Notes in Computer Science*, volume 350. Springer-Verlag, 1989.

[120] T. B. Trafalis and S. Kasap. A novel metaheuristics approach for continuous global optimization. *Journal of Global Optimization*, 23:171–190, 2002.

[121] L. W. Tsai and A. P. Morgan. Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. *Journal of Mechanisms, Transmissions, and Automation in Design*, 107:189–200, 1985.

[122] U. S. Food and Drug Administration. Adverse Event Reporting System. Last time accessed: August 2006. http://www.fda.gov/cder/aers/default.htm.

[123] Uppsala Monitoring Center. Uppsala Monitoring Center - Vigilbase. Last time accessed: August 2006. http://www.umc-products.com/DynPage.aspx?id=4910.

[124] D. Vanderbilt and S. G. Louie. A Monte Carlo simulated annealing approach to optimization over continuous variables. *Journal of Computational Physics*, 56:259–271, 1984.

[125] D. J. Wales and H. A. Scheraga. Global optimization of clusters, crystals, and biomolecules. *Science*, 285:1368–1372, 1999.

[126] D. Whitley, R. Beveridge, C. Graves, and K. Mathias. Test driving three 1995 genetic algorithms: new test functions and geometric matching. *Journal of Heuristics*, 1:77–104, 1995.

[127] Wikipedia. Ackerman Steering Geometry. Last time accessed: August 2006. http://en.wikipedia.org/wiki/Ackermann_steering_geometry.

[128] A. M. Wilson, L. Thabane, and A. Holbrook. Application of data mining in pharmacovigilance. *British Journal of Clinical Pharmacology*, 57(2):127–134, 2003.

[129] J.-M. Yang and C. Y. Kao. A combined evolutionary algorithm for real parameters optimization. In *Proc. of the IEEE International Conference on Evolutionary Computation*, pages 732–737, 1996.

[130] L. Yong, K. Lishan, and D. J. Evans. The annealing evolution algorithm as function optimizer. *Parallel Computing*, 21:389–400, 1993.

[131] H. Yu, J. Yang, W. Wang, and J. Han. Discovering compact and highly discriminative features or feature combinations of drug activities using support vector machines. In *Proc. of the IEEE Bioinformatics Conference*, 2003.

[132] M. Zabarankin, S. Uryasev, and R. Murphey. Aircraft routing under the risk of detection. *To appear in Naval Research Logistics*, 2006.

[133] L. Zhan. Fast stochastic global optimization methods and their applications to cluster crystallization and protein docking. *PhD. thesis, University of Waterloo*, 2005.

[134] T. Zhou, L. Chen, Y. Tang, and X. Zhang. Aligning multiple protein structures. *To appear in Journal of Bioinformatics and Computational Biology*, 2006.

BIOGRAPHICAL SKETCH

Michael J. Hirsch was born in Philadelphia, Pennsylvania. He earned his B.A. in mathematics and computer science from West Chester University of Pennsylvania in 1996, and his M.S. in applied mathematics from the University of Delaware in 1998. From 1996 to 2001, he worked for Aerospace Mass Properties Analysis Corporation, researching new ideas, theories, and algorithms in computer vision for the U.S. Navy. From 2001 to the present, he has been employed with Raytheon, Inc., first in their missile systems division in Tucson, Arizona, and currently in their net-centric systems division in St. Petersburg, Florida. In August of 2004, he was awarded a Raytheon Advanced Study Scholarship, to pursue a doctoral degree. He was on paid leave with Raytheon from August of 2004 to December of 2006, working on his doctorate in the Industrial and Systems Engineering department, University of Florida, under the guidance of Dr. Panos M. Pardalos. After completing his Ph.D., he returned to St. Petersburg and his full-time research at Raytheon. His interests include heuristics for continuous global optimization, object recognition and computer vision, as well as all areas of data and information fusion.