Optimization Letters
DOI 10.1007/s11590-006-0021-6

ORIGINAL PAPER

Global optimization by continuous grasp

M. J. Hirsch · C. N. Meneses · P. M. Pardalos · M. G. C. Resende

Accepted: 25 May 2006

Abstract We introduce a novel global optimization method called *Continuous GRASP* (C-GRASP) which extends Feo and Resende's greedy randomized adaptive search procedure (GRASP) from the domain of discrete optimization to that of continuous global optimization. This stochastic local search method is simple to implement, is widely applicable, and does not make use of derivative information, thus making it a well-suited approach for solving global optimization problems. We illustrate the effectiveness of the procedure on a set of standard test problems as well as two hard global optimization problems.

1 Introduction

Optimization problems arise in numerous settings, including decision-making, engineering, and science [24]. In many situations, convexity of the objective function (or the feasible domain) cannot be easily verified, and it is reasonable

M. J. Hirsch (✉)
Raytheon, Inc. Network Centric Systems, P.O. Box 12248, St. Petersburg, FL 33733-2248, USA
e-mail: mjh8787@ufl.edu

M. J. Hirsch · C. N. Meneses · P. M. Pardalos
Department of Industrial and Systems Engineering, University of Florida, 303 Weil Hall, Gainesville, FL 32611, USA
e-mail: claudio@ufl.edu

P. M. Pardalos
e-mail: pardalos@ufl.edu

M. G. C. Resende
Algorithms and Optimization Research Department, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932, USA
e-mail: mgcr@research.att.com

to assume that multiple local optima exist. Global optimization [16,15] deals with optimization problems with multiple extremal solutions. Global optimization problems can be discrete or continuous. Mathematically, global minimization (optimization) seeks 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 $f : S \to \mathbb{R}$. Such a solution x^* is called a *global minimum*. A solution x' is a *local minimum* in a local neighborhood $S_0 \subset S$ if $f(x') \leq f(x),\ \forall\, x \in S_0$. Global optimization problems abound in many fields, including materials science [26], biology, chemistry, and genetics [4,33], military science [20,19], electrical engineering [11,27], robotics [17,31], and transportation science [24,35].

In this paper, we introduce a novel global optimization method called *Continuous-GRASP* (C-GRASP), which extends the greedy randomized adaptive search procedure (GRASP) of Feo and Resende [7,6,8,28] from the domain of discrete optimization to that of continuous global optimization. Heuristics for global optimization have been previously proposed in the literature, e.g. [2, 3,5,12,14,29,30,32,34]. Our aim 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 global optimization problems. We illustrate the effectiveness of the procedure on a set of standard test problems as well as two hard global optimization problems.

The paper is organized as follows. In Sect. 2, we describe C-GRASP. In Sect. 3, we compare the results of our implementation of C-GRASP with other heuristics from the literature [3,14,29,32] on a set of standard benchmark functions used to test global optimization algorithms. We also show the performance of the C-GRASP heuristic on two real-world problems from the fields of robot kinematics [17,31] and chemical equilibrium systems [21–23]. Final remarks are given in Sect. 4.

2 Continuous GRASP

Feo and Resende [7,6] describe the metaheuristic GRASP as a multistart local search procedure, where each GRASP iteration consists of two phases, a construction phase and a local search phase. Construction combines greediness and randomization to produce a diverse set of good-quality solutions from which to start local search. The best solution over all iterations is kept as the final solution. GRASP has been previously applied to numerous discrete combinatorial optimization problems [8]. This paper describes its first application to continuous global optimization.

In this section, we present the C-GRASP metaheuristic for solving continuous global optimization problems subject to box constraints and describe the construction and local improvement phases. Without loss of generality, we take the domain S as the hyper-rectangle $S = \{x = (x_1, \ldots, x_n) \in \mathbb{R}^n : \ell \leq x \leq u\}$, 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

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

Fig. 1 Pseudo-code for C-GRASP

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

C-GRASP resembles GRASP in that it is a multistart stochastic search metaheuristic that uses a randomized greedy procedure to generate starting solutions for a local improvement algorithm. The main difference is that an iteration of C-GRASP does not consist of a single greedy randomized construction followed by local improvement, but rather a series of construction-local improvement cycles with the output of construction serving as the input of the local improvement, as in GRASP, and the output of the local improvement serving as the input of the construction procedure, unlike GRASP.

Pseudo-code for C-GRASP is shown in Fig. 1. 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 multistart procedure. It is repeated `NumTimesToRun` times in the loop from line 2 to line 16. At each iteration, in line 3, an initial solution x is set to a random point 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. The construction and local improvement phases are then called `MaxIters` times, in sequence, in lines 5 and 6, respectively.

The new solution found after local improvement is compared against the current best solution in line 7. If the new solution has a smaller objective value than the current best solution, then, in line 8, 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 10, `NumIterNoImprov`

Fig. 2 Pseudo-code for C-GRASP construction phase

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

is increased by one. If `NumIterNoImprov` becomes larger than `MaxNumIterNoImprov`, the grid density is increased in line 13, where h is halved and `NumIterNoImprov` is reset to 0. This 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, over all `NumTimesToRun` iterations, is returned.

Procedures `ConstructGreedyRandomized` and `LocalImprovement` called from procedure `C-GRASP` are described in pseudo-codes. The construction phase (see pseudo-code in Fig. 2) 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 6 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 ith coordinate that minimizes the objective function is saved, as well as the objective function value g_i in lines 6 and 7 of the pseudo-code. After the line search is performed for each of the unfixed coordinates, in lines 12–17 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 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 18 and 19 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

Fig. 3 Pseudo-code for C-GRASP local improvement phase

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

all of the n coordinates of x have been fixed. At that stage, x is returned from the construction phase.

The local improvement phase (with pseudo-code shown in Fig. 3) 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 is not efficiently computable for all functions. 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.

For a problem in two dimensions (and easily generalized to n dimensions), given a point x, the eight possible directions to be 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, `Ternary`$(15) = 120$. If we look at a modified `Ternary` function, say `Ternary′`, where each $r \in \mathbb{N}$ gets mapped to its ternary representation, with each '2' being replaced by a '-1' (e.g. `Ternary′`$(15) = \{1,-1,0\}$), then the function `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 `Ternary′` with input a value $r \in \{1,\ldots,3^n-1\}$. The output will be the appropriate direction vector. Table 1 gives an example of the `Ternary′` mapping when $n = 2$.

Table 1 Ternary′ mapping ($n = 2$)

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

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 2. As seen above, for even moderate values of n, the number of possible search directions can be quite large. To keep the local improvement tractable, we set the variable `NumDirToTry` equal to the minimum of $3^n - 1$ and a user-defined value `MaxDirToTry` in line 3. Starting at the point x^*, in the loop from line 6 to line 17, we construct up to `NumDirToTry` distinct random directions, in turn. In line 9, direction d is constructed and the test point $x = x^* + h * d$ is computed, where h is the parameter defined earlier that controls the density of the discretization. If the test point 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.

3 Computational Experiments

We report on computational testing of the C-GRASP heuristic. First, we describe our test environment. Then, we compare our implementation of C-GRASP with other global optimization heuristics on a set of standard test functions. Finally, we show the performance of C-GRASP on two real-world problems from the fields of robot kinematics and chemical equilibrium systems.

3.1 Test environment

All 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. The C-GRASP heuristic was implemented 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`. CPU times were computed using the function `getusage()`.

The algorithm used for random-number generation is an implementation of the multiplicative linear congruential generator [25], with parameters 16807 (multiplier) and $2^{31} - 1$ (prime number).

Table 2 C-GRASP parameter values

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

C-GRASP has six parameters that need to be set. Parameter α is used to set up the restricted candidate list. `MaxDirToTry` is the maximum number of directions searched in the local improvement phase. For a fixed search space discretization, `MaxNumIterNoImprov` is the maximum number of calls to the local improvement procedure with no improvement. Parameter h is the initial search space discretization value. Parameter `MaxIters` is the maximum number of construction-local improvement cycles per major iteration and parameter `NumTimesToRun` is the maximum number of multistart, or major, iterations. Unless otherwise stated, the parameters for all C-GRASP runs are listed in Table 2.

3.2 Comparing C-GRASP with other heuristics

We applied our implementation of C-GRASP to a set of 14 standard continuous global optimization test problems found in the literature. These test problems are from eight classes of problem instances: Branin [2,3,5,13,14,18, 30,32,34], Eason [12–14], Goldstein-Price [2,3,5,9,12–14,29,30,32], Shubert [2,5,12–14,18,30,34], Hartmann [2,3,5,12–14,29,32], Rosenbrock [12–14,29, 30,34], Shekel [3,12–14,29,30,32], and Zakharov [12–14,29]. 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 [14], "the characteristics of these test functions are diverse enough to cover many kinds of difficulties that arise in global optimization problems."

Since the global minimum is known for each of these functions, the C-GRASP heuristic was run until the objective function value $\tilde{f}$ was significantly close to the global optimum f^* or `NumTimesToRun` restarts were done. As in [12–14, 29], we define *significantly close* by

$$|f^* - \tilde{f}| \leq \epsilon_1 |f^*| + \epsilon_2, \tag{1}$$

where $\epsilon_1 = 10^{-4}$ and $\epsilon_2 = 10^{-6}$.

For each problem, we made 100 independent C-GRASP runs. Each run used a different starting seed for the pseudo-random number generator. We recorded the number of function evaluations and the elapsed time for the current solution to satisfy stopping criterion 1. To test robustness of the heuristic, we computed the percentage of runs in which it finds a solution satisfying Eq. 1 in less than `NumTimesToRun` restarts. We also computed the

Table 3 Heuristics used in testing

Method	Reference
Enhanced simulated annealing (ESA)	[29]
Monte Carlo simulated annealing (MCSA)	[32]
Sniffer global optimization (SGO)	[3]
Directed tabu search (DTS)	[14]
Continuous GRASP (C-GRASP)	This article

Table 4 Summary of results for heuristics on test functions

Function	ESA	MCSA	SGO	DTS	C-GRASP
Branin	–	100/557	100/205	100/212	100/59857/0.0016
Easom	–	–	–	82/223	100/89630/0.0042
Goldstein-Price	100/783	99/1186	100/664	100/230	100/29/0.0000
Shubert	–	–	–	92/274	100/82363/0.0078
Hartmann-3	100/698	100/1224	99/534	100/438	100/20743/0.0026
Hartmann-6	100/1638	62/1914	99/1760	83/1787	100/79685/0.0140
Rosenbrock-2	–	–	–	100/254	100/1158350/0.01320
Rosenbrock-5	–	–	–	85/1684	100/6205503/1.7520
Rosenbrock-10	–	–	–	85/9037	99/20282529/11.4388
Shekel-(4,5)	54/1487	54/3910	90/3695	75/819	100/5545982/2.3316
Shekel-(4,7)	54/1661	64/3421	96/2655	65/812	100/4052800/2.3768
Shekel-(4,10)	50/1363	81/3078	95/3070	52/828	100/4701358/3.5172
Zakharov-5	–	–	–	100/1003	100/959/0.0000
Zakharov-10	–	–	–	100/4032	100/3607653/1.0346

In all heuristic columns, the first entry refers to percentage of runs significantly close to optimum, the second entry refers to average number of function evaluations, and (for C-GRASP) the third entry refers to average time, in CPU seconds to find a solution significantly close to optimum

average time needed for the algorithm to find such a solution.[1] In comparing the C-GRASP with those previously published, we accepted those published results as valid (i.e., we did not program those algorithms ourselves). The same measure of closeness used in the stopping criterion of C-GRASP was used for the enhanced simulated annealing (ESA) [29] and directed tabu search (DTS) [14]. However, the Monte Carlo simulated annealing (MCSA) [32] and the Sniffer global optimization (SGO) [3] heuristics used Eq. 1 in a slightly different form, with $\epsilon_1 = 10^{-3}$ and $\epsilon_2 = 0$.

Table 3 lists the heuristics used in the comparison. We chose these heuristics because each was tested with most, if not all, of the test functions considered in this paper. Table 4 displays the results of the heuristics for the functions on which they were tested (a '–' in the table indicates that a function was not used in testing a heuristic). For each test function/heuristic pair, the first entry

[1] In the single C-GRASP run where the closeness criterion was not met in fewer than `NumTimesToRun` restarts, we used the total time of the run in the average time computation.

denotes the percentage of runs significantly close to the global optimum, the second entry denotes the average number of function evaluations, and for the C-GRASP heuristic, the third entry is the average time (in seconds) needed to find a solution satisfying termination criterion 1.

With respect to percentage of runs significantly close to the global optimum, it is clear that C-GRASP outperforms the other heuristics. It is understandable that C-GRASP, on average, performs more function evaluations than the other algorithms – the C-GRASP construction phase performs a high number of function evaluations, and the C-GRASP heuristic uses neither a priori information about the functions, nor derivative information. As stated in [1], "In a black box situation where (virtually) no a priori information is available about the structure of the optimization problem, a large number of function evaluations is required before the location of the global optimum can be specified with confidence." However, despite the high number of function evaluations, Table 4 shows that the C-GRASP heuristic is still quite fast, i.e., taking at most a few seconds to converge.

3.3 Robot kinematics application

We consider next a problem from robot kinematics [10,17,31]. We are given a six-revolute manipulator (rigid bodies, or links, connected together by joints, with each link connected to no more than two others), with the first link designated the base, and the last link designated the hand of the robot. The problem is to determine the possible positions of the hand, given that the joints are movable. In [31], this problem is reduced to solving a system of eight nonlinear equations in eight unknowns. While this might seem like a simple task, Floudas et al. [10] state that "this is a challenging problem." Let us write the unknowns as $x = \{x_1, \ldots, x_8\} \in [-1,1]^8$ and the equations as $f_1(x), \ldots, f_8(x)$ (the interested reader is directed to [10,17] for the actual equations). With this system, we form the optimization problem

$$\text{Find } x^* = \operatorname{argmin}\{F(x) = \sum_{i=1}^{8} f_i^2(x) \mid x \in [-1,1]^8\}. \tag{2}$$

Since $F(x) \geq 0$ for all $x \in [-1,1]^8$, it is easy to see that $F(x) = 0 \iff f_i(x) = 0$ for all $i \in \{1, \ldots, 8\}$. Hence, we have the following: $\exists\, x^* \in [-1,1]^8 \ni F(x^*) = 0 \implies x^*$ is a global minimizer of problem 2 and x^* is a root of the system of equations $f_1(x), \ldots, f_8(x)$. From [10,17], in the given domain, there are 16 known roots to this system. However, solving problem (2) 16 times using C-GRASP (or any heuristic, for that matter) with different starting solutions gives no guarantee of finding all 16 roots. It is entirely possible that some of the roots would be found multiple times, while others would not be found at all.

To avoid this, we modified the objective function $F(x)$. Suppose that heuristic has just found the kth root (roots are denoted $x^1, \ldots, x^k$). Then the main C-GRASP will restart, with the modified objective function given by

$$F(x) = \sum_{i=1}^{8} f_i^2(x) + \beta \sum_{j=1}^{k} \mathrm{e}^{-\|x-x^j\|} \chi_\rho(\|x - x^j\|), \tag{3}$$

where

$$\chi_\rho(\delta) = \begin{cases} 1 & \text{if } \delta \leq \rho \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

β is a large constant, and ρ is a small constant. This has the effect of creating an area of repulsion near solutions that have already been found by the heuristic.

For this problem, we ran C-GRASP ten times (a different starting random number seed for each run) with $\rho = 1, \beta = 10^{10}$, and `MaxItersNoImprov` $= 5$. In each case, the heuristic was able to find all 16 known roots. The average CPU time needed to find the 16 roots was 3,048 s. We note that the aim of this example is not to compare our approach with the one in [17], which makes use of the gradient. Rather, we aim to show that for a challenging problem, where derivative information can either be unavailable or too expensive to compute, C-GRASP is robust enough to find the solution.

3.4 Chemical equilibrium systems

We now consider interactions and equilibrium of chemical systems. More specifically, we examine the combustion of propane (C_3H_8) in air (O_2 and N_2). Meintjes and Morgan [23] provide the derivation of this chemical reaction. This problem produces a system of ten nonlinear equations in ten unknowns. There is one physical solution to this system in which all the variables are positive. Due to the difficulty in finding this solution,Meintjes and Morgan [21–23] derive a transformation to place the system in canonical form. The canonical form is a system of five nonlinear equations in five unknowns.

For both systems, we formed an objective function as the sum of the squares of the nonlinear equations. We ran the C-GRASP heuristic ten times (each run with a different starting random number seed), with the parameter `MaxNumIterNoImprov` set to 10. For the system in canonical form, C-GRASP was successful on each run, i.e., the solution found had an objective value satisfying Eq.1, while for the more difficult original system, C-GRASP was successful on eight of the ten runs. The average time for the canonical runs was 37.53 s, while for the original system, C-GRASP took 201.58 s, on average. Meintjes and Morgan [23] solve the canonical problem by using a variant of Newton's method (absolute Newton's method), which requires the gradient of each equation in the system. They did not report their success on solving the original, more difficult system.

4 Concluding remarks

In this paper, we introduced a new metaheuristic called continuous GRASP, or C-GRASP, for continuous global optimization. This is the first time the GRASP metaheuristic for discrete combinatorial optimization has been modified for the continuous setting. As seen in Sect. 2, the algorithm is easy to describe and implement, whether on a single machine, or in a parallel architecture, and is generally applicable to global optimization problems. In addition, it makes use of neither derivative nor a priori information, making it an ideal solution method for *black-box* problems.

While sometimes requiring many function evaluations, Sect. 3 shows that for a set of standard test functions, the C-GRASP heuristic almost always converges to the global optimum in a very short amount of time. This section also shows the ability of C-GRASP to solve more challenging problems with real-world applications, thus making it a well-suited approach for solving global optimization problems from many fields of the physical sciences.

Acknowledgements AT&T Labs Research Technical Report: TD-6MPUV9. This work has been partially supported by NSF, NIH, and CRDF grants. C.N. Meneses was supported in part by the Brazilian Federal Agency for Higher Education (CAPES) – Grant No. 1797-99-9.

References

1. Anderssen, R.S.: Global optimization. In: Anderssen, R.S., Jennings, L.S., Ryan, D.M. (eds.) Optimization, pp. 26–48. Queensland University Press, (1972)
2. Barhen, J., Protopopescu, V., Reister, D.: Trust: a deterministic algorithm for global optimization. Science, **276**, 1094–1097 (1997)
3. Butler, R.A.R., Slaminka, E.E.: An evaluation of the sniffer global optimization algorithm using standard test functions. J. Comput. Phys. **99**(1), 28–32 (1992)
4. Chen, L., Zhou, T., Tang, Y.: Protein structure alignment by deterministic annealing. Bioinformatics **21**(1), 51–62 (2005)
5. Cvijović D., Klinowski J.: Taboo search: an approach to the multiple minima problem. Science **267**, 664–666 (1995)
6. Feo, T.A., Resende, M.G.C.: A probabilistic heuristic for a computationally difficult set covering problem. Oper. Res. Lett. **8**, 67–71 (1989)
7. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedures. J. Global. Optim. **6**, 109–133 (1995)
8. Festa, P., Resende, M.G.C.: GRASP: an annotated bibliography. In: Ribeiro, C.C., Hansen, P., (eds.) Essays and Surveys in Metaheuristics, pp. 325–367. Kluwer, Dordrecht (2002)
9. Floudas, C.A., Pardalos P.M.: A collection of test problems for constrained global optimization algorithms. In: Goods, G., Hartmanis, J., (eds.) Lecture Notes in Computer Science, vol. 455. Springer Berlin Heidelberg New York (1990)
10. Floudas, C.A., Pardalos, P.M., Adjiman, C., Esposito, W., Gumus, Z., Harding, S., Klepeis, J., Meyer, C., Schweiger C.: Handbook of Test Problems in Local and Global Optimization. Kluwer, Dordrecht (1999)
11. Granvilliers, L., Benhamou, F.: Progress in the solving of a circuit design problem. J. Global Optim. **20**(2), 155–168 (2001)
12. Hedar, A.R., Fukushima, M.: Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization. Optim. Methods Softw. **17**, 891–912 (2002)
13. Hedar, A.R., Fukushima M.: Minimizing multimodal functions by simplex coding genetic algorithms. Optim. Methods Softw. **18**, 265–282, (2003)
14. Hedar, A.R., Fukushima, M.: Tabu search directed by direct search methods for nonlinear global optimization. Eur. J. Oper. Res. **170**, 329–349 (2006)

15. Horst, R., Tuy, H.: Global Optimization: Deterministic Approaches, 2nd Revised Edition. Springer Berlin Heidelberg New York (1993)
16. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to Global Optimization. Kluwer Dordrecht 2nd edn. (2000)
17. Kearfott, R.B.: Some tests of generalized bisection. ACM Trans. Math. Softw. **13**(3), 197–220 (1987)
18. Koon, G.H., Sebald, A.V.: Some interesting test functions for evaluating evolutionary programming strategies. In Proceedings of the Fourth Annual Conference on Evolutionary Programming, pp. 479–499 (1995)
19. Krokhmal, P., Murphey, R., Pardalos, P.M., Uryasev, S., Zrazhevsky, G.: Robust decision making: addressing uncertainties in distributions. In: Butenko, S., Murphey, R., Pardalos, P.M.: (eds.) Cooperative Control: Models, Applications, and Algorithms, pp. 165–185. Kluwer Dordrecht (2003)
20. Krokhmal, P., Murphey, R., Pardalos, P.M., Uryasev, S.: Use of conditional value-at-risk in stochastic programs with poorly defined distributions. In: Butenko, S., Murphey, R., Pardalos, P.M., (eds.) Recent Developments in Cooperative Control and Optimization, pp. 225–243. Kluwer Dordrecht (2004)
21. Meintjes, K., Morgan, A.P.: A methodology for solving chemical equilibrium systems. Appl. Math. Comput. **22**, 333–361 (1987)
22. Meintjes, K., Morgan, A.P.: Element variables and the solution of complex chemical equilibrium problems. Combust. Sci. Technol. **68**, 35–48 (1989)
23. Meintjes, K., Morgan, A.P.: Chemical equilibrium systems as numerical test problems. ACM Trans. Math. Softw. **16**(2), 143–151 (1990)
24. Pardalos, P.M., Resende, M.G.C.: (eds.) Handbook of Applied Optimization. Oxford University Press, New York (2002)
25. Park, S., Miller, K.: Random number generators: good ones are hard to find. Commun ACM **31**, 1192–1201 (1988)
26. Pham, D.T., Karaboga, D.: Intelligent Optimization Techniques: Genetic Algorithms, Tabu Search, Simulated Annealing, and Neural Networks. Springer Berlin Heidelberg New York (2000)
27. Ratschek, H., Rokne, J.: Experiments using interval analysis for solving a circuit design problem. J. Global Optim. **3**(3), 501–518 (1993)
28. Resende, M.G.C., Ribeiro, C.C.: Greedy randomized adaptive search procedures. In: Glover, F., Kochenberger, G. (eds.) Handbook of Metaheuristics, pp. 219–249. Kluwer Dordrecht (2003)
29. Siarry, P., Berthiau, G., Durbin, F., Haussy, J.: Enhanced simulated annealing for globally minimizing functions of many continuous variables. ACM Trans. Math. Softw. **23**(2), 209–228 (1997)
30. Trafalis, T.B., Kasap, S.: A novel metaheuristics approach for continuous global optimization. J. Global Optim. **23**, 171–190 (2002)
31. Tsai, L.W., Morgan, A.P.: Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods. J. Mech. Transm. Autom. Des. **107**, 189–200 (1985)
32. Vanderbilt, D., Louie, S.G.: A Monte Carlo simulated annealing approach to optimization over continuous variables. J. Comput. Phys. **56**, 259–271 (1984)
33. Wales, D.J., Scheraga, H.A.: Global optimization of clusters, crystals, and biomolecules. Science, **285**, 1368–1372 (1999)
34. Yang, J.-M., Kao, C.Y.: A combined evolutionary algorithm for real parameters optimization. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 732–737, (1996)
35. Zabarankin, M., Uryasev, S., Murphey, R.: Aircraft routing under the risk of detection. Naval Res. Logist. (2006), accepted for publication