Global optimization by continuous GRASP

Mauricio G. C. Resende[1] Michael J. Hirsch[2,3]
Claudio N. Meneses[2] Panos M. Pardalos[2]

[1]AT&T Labs Research, Florham Park, New Jersey

[2]University of Florida, Gainesville, Florida

[3]Raytheon, Inc., St. Petersburg, Florida

19th International Symposium on
Mathematical Programming
Rio de Janeiro, Brazil – July 31 to August 4, 2006

Outline

1. Introduction
   - Global Optimization
   - Continuous GRASP
2. Continuous GRASP
   - GRASP
   - C-GRASP
   - Construction and Local Improvement
3. Experimental Results
   - Experiment Setup
   - Comparing with Other Heuristics
   - Real-world applications
4. Conclusion

Outline

1. Introduction
   - Global Optimization
   - Continuous GRASP
2. Continuous GRASP
   - GRASP
   - C-GRASP
   - Construction and Local Improvement
3. Experimental Results
   - Experiment Setup
   - Comparing with Other Heuristics
   - Real-world applications
4. Conclusion

Outline

1. Introduction
   - Global Optimization
   - Continuous GRASP
2. Continuous GRASP
   - GRASP
   - C-GRASP
   - Construction and Local Improvement
3. Experimental Results
   - Experiment Setup
   - Comparing with Other Heuristics
   - Real-world applications
4. Conclusion

Outline

1. Introduction
   - Global Optimization
   - Continuous GRASP
2. Continuous GRASP
   - GRASP
   - C-GRASP
   - Construction and Local Improvement
3. Experimental Results
   - Experiment Setup
   - Comparing with Other Heuristics
   - Real-world applications
4. Conclusion

Global Optimization

- Optimization problems arise in numerous settings, e.g. decision-making, engineering.
- Global optimization (GO) are optimization problems with multiple extremal solutions.
- GO problems can be discrete or continuous.

Global Optimization

- Optimization problems arise in numerous settings, e.g. decision-making, engineering.
- Global optimization (GO) are optimization problems with multiple extremal solutions.
- GO problems can be discrete or continuous.

Global Optimization

- Optimization problems arise in numerous settings, e.g. decision-making, engineering.
- Global optimization (GO) are optimization problems with multiple extremal solutions.
- GO problems can be discrete or continuous.

Global Optimization Problem

- GO (minimization) 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 $f : S \rightarrow \mathbb{R}$.
- 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 Problem

- GO (minimization) 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 $f : S \rightarrow \mathbb{R}$.
- 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 Problem

- GO (minimization) 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 $f : S \rightarrow \mathbb{R}$.
- 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 $f(x') \leq f(x),\ \forall\, x \in S_0$.

Continuous GRASP

- Continuous-GRASP (C-GRASP) extends the greedy randomized adaptive search procedure (GRASP) of Feo and Resende (1989, 1995) from the domain of discrete optimization to that of continuous global optimization.
- C-GRASP is 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.
- We illustrate the effectiveness of the procedure on a set of standard test problems as well as two hard global optimization problems.

Continuous GRASP

- Continuous-GRASP (C-GRASP) extends the greedy randomized adaptive search procedure (GRASP) of Feo and Resende (1989, 1995) from the domain of discrete optimization to that of continuous global optimization.
- C-GRASP is 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.
- We illustrate the effectiveness of the procedure on a set of standard test problems as well as two hard global optimization problems.

Continuous GRASP

- Continuous-GRASP (C-GRASP) extends the greedy randomized adaptive search procedure (GRASP) of Feo and Resende (1989, 1995) from the domain of discrete optimization to that of continuous global optimization.
- C-GRASP is 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.
- We illustrate the effectiveness of the procedure on a set of standard test problems as well as two hard global optimization problems.

GRASP

- GRASP is a multi-start 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 (Festa & Resende, 2001).

GRASP

- GRASP is a multi-start 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 (Festa & Resende, 2001).

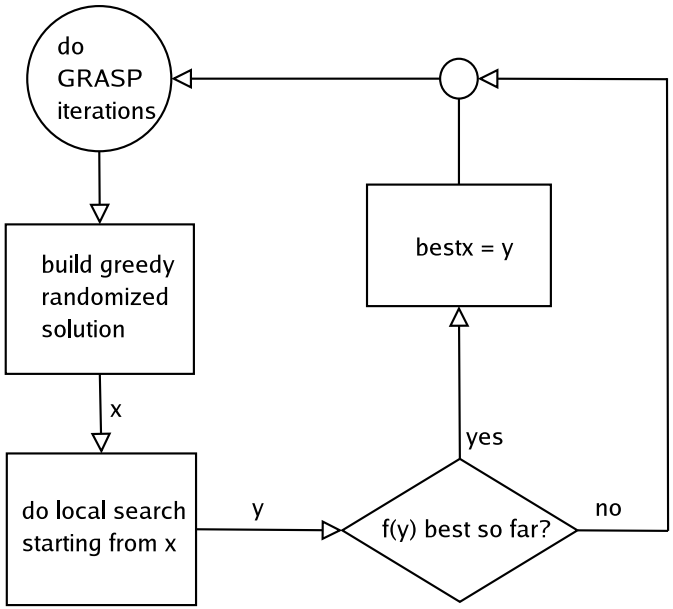
GRASP

- GRASP is a multi-start 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 (Festa & Resende, 2001).

GRASP

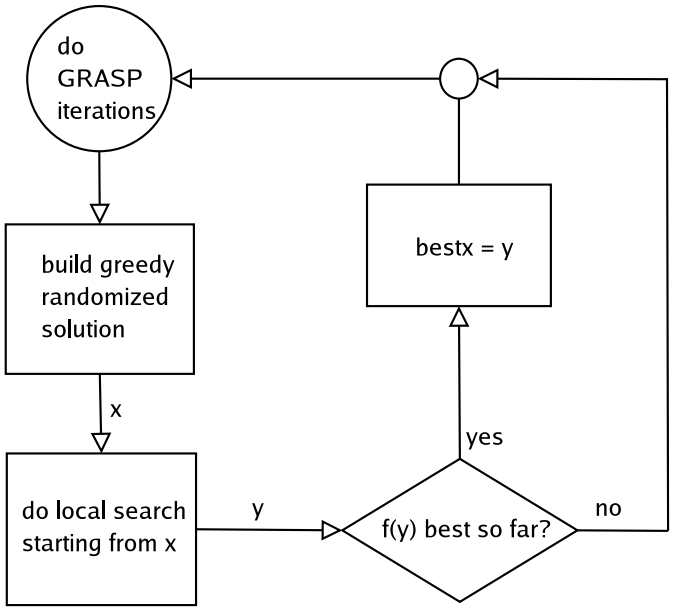
- GRASP is a multi-start local search procedure, where each GRASP iteration consists of two phases, a construction phase and a local search phase.
- Construction combines greediness and randomization to produce a diverse set of good-quality solutions from which to start local search.
- The best solution over all iterations is kept as the final solution.
- GRASP has been previously applied to numerous discrete combinatorial optimization problems (Festa & Resende, 2001).

GRASP



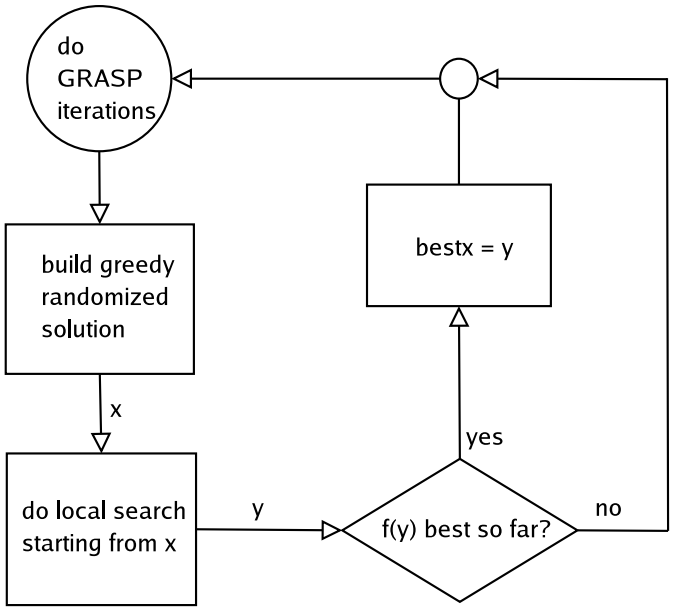
- Multi-start stochastic search.
- Each iteration:
  - Construct greedy randomized solution x.
  - Perform local search starting from x.
- Best solution is returned.

GRASP



- Multi-start stochastic search.
- Each iteration:
  - Construct greedy randomized solution x.
  - Perform local search starting from x.
- Best solution is returned.

GRASP



- Multi-start stochastic search.
- Each iteration:
  - Construct greedy randomized solution x.
  - Perform local search starting from x.
- Best solution is returned.

Continuous GRASP.

- C-GRASP is a metaheuristic for solving continuous global optimization problems subject to box constraints.
- Without loss of generality, we take the domain S as the hyper-rectangle $S = \{x = (x_1, \ldots, x_n) \in \mathbb{R}^n : l \leq x \leq u\}$, where $l \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$ such that $u_i \geq l_i$, for $i = 1, \ldots, n$.
- The global optimization problem considered here is to find

$$x^* = \text{argmin}\{f(x) \mid l \leq x \leq u\},$$

where $f : \mathbb{R}^n \to \mathbb{R}$, and $l, x, u \in \mathbb{R}^n$.

Continuous GRASP.

- C-GRASP is a metaheuristic for solving continuous global optimization problems subject to box constraints.
- Without loss of generality, we take the domain S as the hyper-rectangle $S = \{x = (x_1, \ldots, x_n) \in \mathbb{R}^n : l \leq x \leq u\}$, where $l \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$ such that $u_i \geq l_i$, for $i = 1, \ldots, n$.
- The global optimization problem considered here is to find

$$x^* = \text{argmin}\{f(x) \mid l \leq x \leq u\},$$

where $f : \mathbb{R}^n \to \mathbb{R}$, and $l, x, u \in \mathbb{R}^n$.

Continuous GRASP.

- C-GRASP is a metaheuristic for solving continuous global optimization problems subject to box constraints.
- Without loss of generality, we take the domain S as the hyper-rectangle $S = \{x = (x_1, \ldots, x_n) \in \mathbb{R}^n : l \leq x \leq u\}$, where $l \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$ such that $u_i \geq l_i$, for $i = 1, \ldots, n$.
- The global optimization problem considered here is to find

$$x^* = \text{argmin}\{f(x) \mid \ l \leq x \leq u\},$$

where $f : \mathbb{R}^n \to \mathbb{R}$, and $l, x, u \in \mathbb{R}^n$.

Continuous GRASP

- C-GRASP resembles GRASP in that it is a multi-start 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.
  - Instead, it consists of a series of construction-local improvement cycles where, as in GRASP, the output of construction is the input of the local improvement.
  - Unlike GRASP, the output of the local improvement is the input of the construction procedure.

Continuous GRASP

- C-GRASP resembles GRASP in that it is a multi-start 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.
  - Instead, it consists of a series of construction-local improvement cycles where, as in GRASP, the output of construction is the input of the local improvement.
  - Unlike GRASP, the output of the local improvement is the input of the construction procedure.

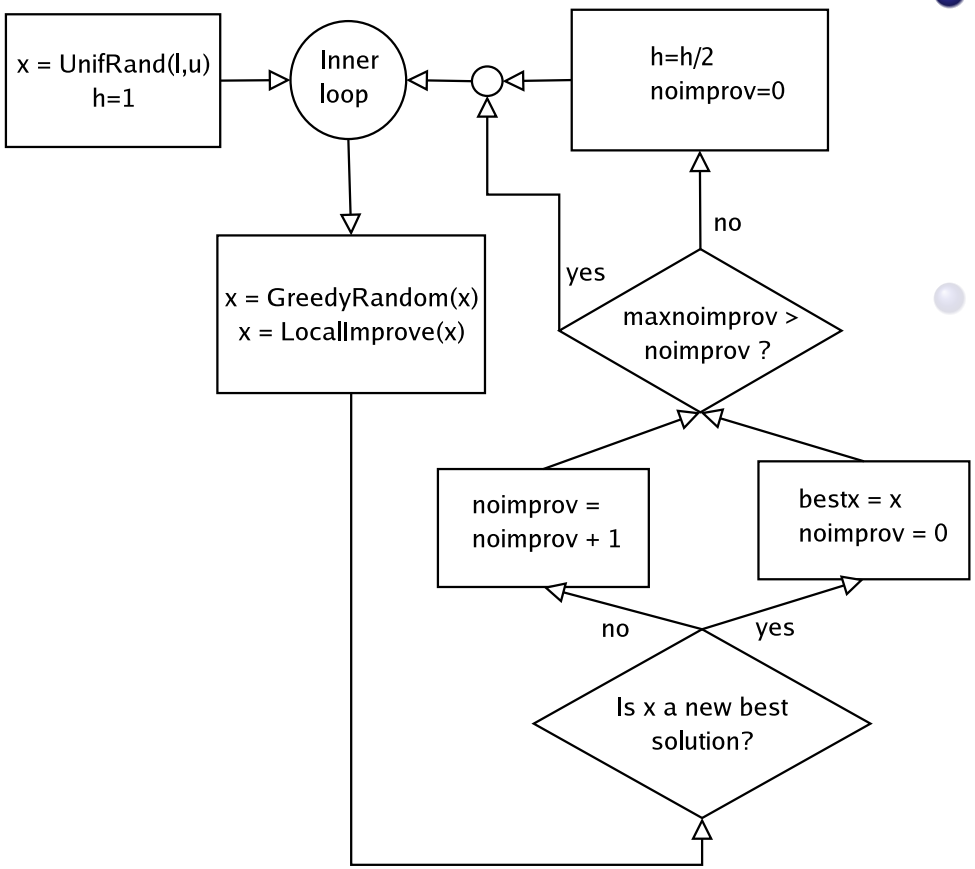
Continuous GRASP

- C-GRASP resembles GRASP in that it is a multi-start 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.
  - Instead, it consists of a series of construction-local improvement cycles where, as in GRASP, the output of construction is the input of the local improvement.
  - Unlike GRASP, the output of the local improvement is the input of the construction procedure.

Continuous GRASP

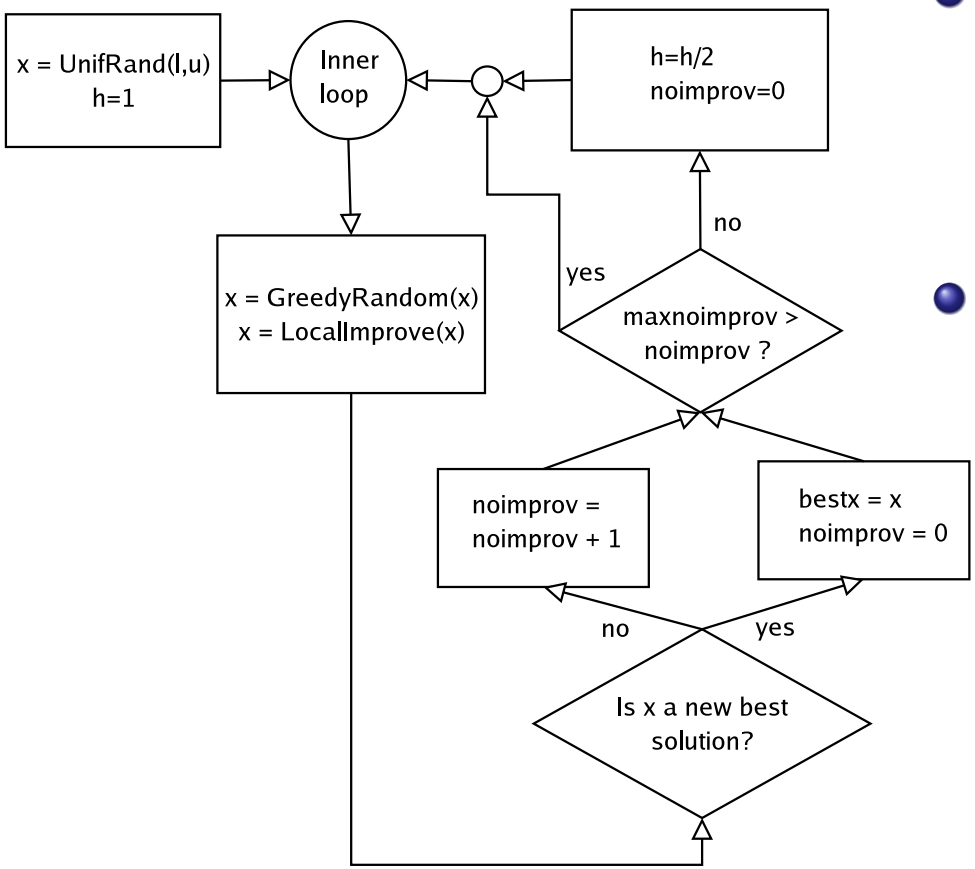
- C-GRASP resembles GRASP in that it is a multi-start stochastic search metaheuristic that uses a randomized greedy procedure to generate starting solutions for a local improvement algorithm.
- The main difference is that
  - An iteration of C-GRASP does not consist of a single greedy randomized construction followed by local improvement.
  - Instead, it consists of a series of construction-local improvement cycles where, as in GRASP, the output of construction is the input of the local improvement.
  - Unlike GRASP, the output of the local improvement is the input of the construction procedure.

C-GRASP: Major iteration of multi-start procedure



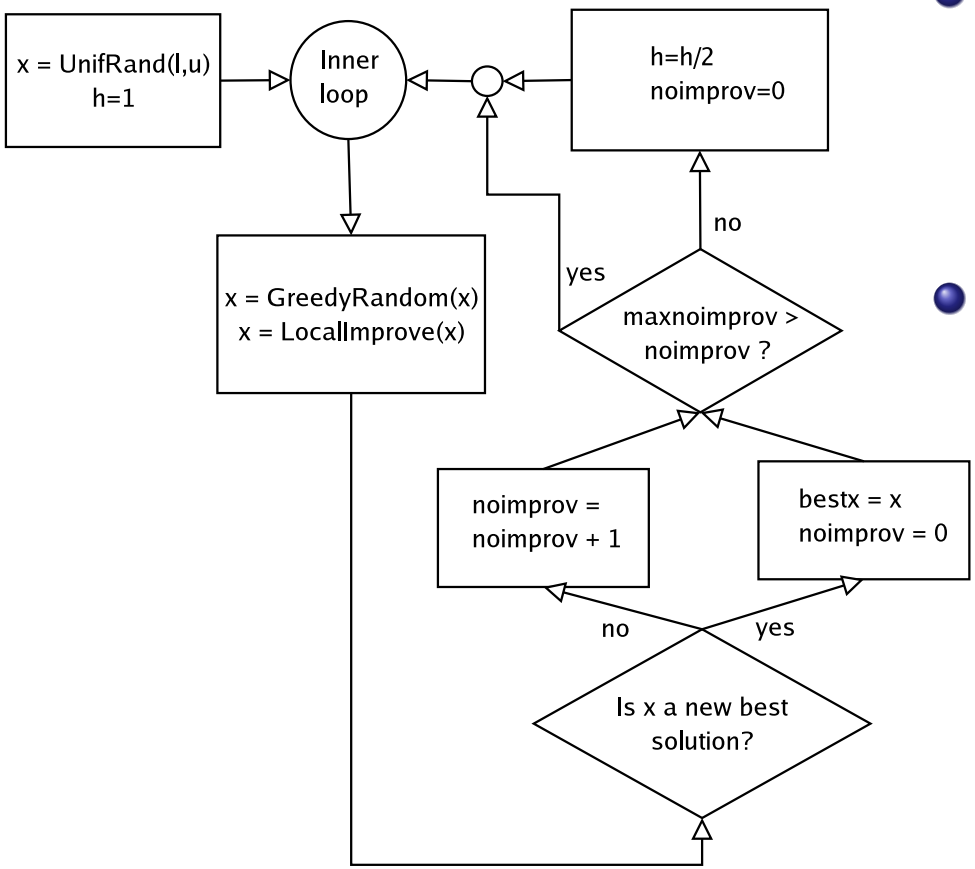
- Each major iteration consists of a fixed number of minor iterations.
- Each minor iteration:
  - Construct greedy randomized solution.
  - Attempt local improvement.
  - Adjust search space discretization *h* if necessary.

C-GRASP: Major iteration of multi-start procedure



- Each major iteration consists of a fixed number of minor iterations.
- Each minor iteration:
  - Construct greedy randomized solution.
  - Attempt local improvement.
  - Adjust search space discretization *h* if necessary.

C-GRASP: Major iteration of multi-start procedure



- Each major iteration consists of a fixed number of minor iterations.
- Each minor iteration:
  - Construct greedy randomized solution.
  - Attempt local improvement.
  - Adjust search space discretization *h* if necessary.

Construction phase

- Construction starts from x.
- Initialize coordinates set $S \leftarrow \{1, 2, \ldots, n\}$.
- While $S \neq \emptyset$ do:
  - Let $z_i \leftarrow LineSearch(x, f(), i)$ and $f_i = f(z_i)$. Let $f_{\max}$ and $f_{\min}$ be the f values of the best and worst directions in S, respectively.
  - Set $RCL = \{i \in S \mid f_i \leq (1 - \alpha) \cdot f_{\min} + \alpha \cdot f_{\max}\}$ where α is a parameter such that $0 \leq \alpha \leq 1$.
  - Select index j at random from RCL and set $x_j \leftarrow z_j$ and $S \leftarrow S \setminus \{j\}$
- Return x.

Local Improvement

- C-GRASP makes no use of gradients.
- Local improvement phase can be seen as approximating role of gradient.
- From a given input point $x^* \in \mathbb{R}^n$, the local improvement generates a set of directions and determines in which direction, if any, the objective function value improves.
- For direction d, test solution is $x \leftarrow x^* + h * d$, where h is the search space discretization parameter.
- If $l \leq x \leq u$ and $f(x) < f(x^*)$, then procedure moves to x and $x^* \leftarrow x$.

Local Improvement

- C-GRASP makes no use of gradients.
- Local improvement phase can be seen as approximating role of gradient.
- From a given input point $x^* \in \mathbb{R}^n$, the local improvement generates a set of directions and determines in which direction, if any, the objective function value improves.
- For direction d, test solution is $x \leftarrow x^* + h * d$, where h is the search space discretization parameter.
- If $l \leq x \leq u$ and $f(x) < f(x^*)$, then procedure moves to x and $x^* \leftarrow x$.

Local Improvement

- C-GRASP makes no use of gradients.
- Local improvement phase can be seen as approximating role of gradient.
- From a given input point $x^* \in \mathbb{R}^n$, the local improvement generates a set of directions and determines in which direction, if any, the objective function value improves.
- For direction d, test solution is $x \leftarrow x^* + h * d$, where h is the search space discretization parameter.
- If $l \leq x \leq u$ and $f(x) < f(x^*)$, then procedure moves to x and $x^* \leftarrow x$.

Local Improvement

- C-GRASP makes no use of gradients.
- Local improvement phase can be seen as approximating role of gradient.
- From a given input point $x^* \in \mathbb{R}^n$, the local improvement generates a set of directions and determines in which direction, if any, the objective function value improves.
- For direction d, test solution is $x \leftarrow x^* + h * d$, where h is the search space discretization parameter.
- If $l \leq x \leq u$ and $f(x) < f(x^*)$, then procedure moves to x and $x^* \leftarrow x$.

Local Improvement

- C-GRASP makes no use of gradients.
- Local improvement phase can be seen as approximating role of gradient.
- From a given input point $x^* \in \mathbb{R}^n$, the local improvement generates a set of directions and determines in which direction, if any, the objective function value improves.
- For direction d, test solution is $x \leftarrow x^* + h * d$, where h is the search space discretization parameter.
- If $l \leq x \leq u$ and $f(x) < f(x^*)$, then procedure moves to x and $x^* \leftarrow x$.

Local Improvement Directions

- Directions are generated at random. Repetitions are not allowed.
- A maximum number of directions to be generated is an input parameter.
- We use a function $T'(i)$ that maps the integers

$$i \in \{1, 2, \ldots, 3^n - 1\}$$

to the directions

$$T'(i) \in d_1, d_2, \ldots, d_{3^n-1},$$

where each coordinate of d_i is one of $\{-1, 0, 1\}$.

Local Improvement Directions

- Directions are generated at random. Repetitions are not allowed.
- A maximum number of directions to be generated is an input parameter.
- We use a function $T'(i)$ that maps the integers

$$i \in \{1, 2, \ldots, 3^n - 1\}$$

to the directions

$$T'(i) \in d_1, d_2, \ldots, d_{3^n-1},$$

where each coordinate of d_i is one of $\{-1, 0, 1\}$.

Local Improvement Directions

- Directions are generated at random. Repetitions are not allowed.
- A maximum number of directions to be generated is an input parameter.
- We use a function $T'(i)$ that maps the integers

$$i \in \{1, 2, \ldots, 3^n - 1\}$$

to the directions

$$T'(i) \in d_1, d_2, \ldots, d_{3^n-1},$$

where each coordinate of d_i is one of $\{-1, 0, 1\}$.

Local Improvement Directions

i	$\longrightarrow$	$\mathrm{T}(i)$	$\longrightarrow$	$\mathrm{T}'(i)$	i	$\longrightarrow$	$\mathrm{T}(i)$	$\longrightarrow$	$\mathrm{T}'(i)$
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	$\longrightarrow$	$\{-1,-1\}$

- $T(i)$ is the base-3 representation of i.
- $T'(i)$ is $T(i)$ with all 2's replaced by -1's.

Local Improvement Directions

i	$\longrightarrow$	$\mathrm{T}(i)$	$\longrightarrow$	$\mathrm{T}'(i)$	i	$\longrightarrow$	$\mathrm{T}(i)$	$\longrightarrow$	$\mathrm{T}'(i)$
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	$\longrightarrow$	$\{-1,-1\}$

- $T(i)$ is the base-3 representation of i.
- $T'(i)$ is $T(i)$ with all 2's replaced by -1's.

Local Improvement Directions

i	$\longrightarrow$	$\mathrm{T}(i)$	$\longrightarrow$	$\mathrm{T}'(i)$	i	$\longrightarrow$	$\mathrm{T}(i)$	$\longrightarrow$	$\mathrm{T}'(i)$
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	$\longrightarrow$	$\{-1,-1\}$

- $T(i)$ is the base-3 representation of i.
- $T'(i)$ is $T(i)$ with all 2's replaced by -1's.

Experimental Results

- Compare C-GRASP with other global optimization heuristics on a set of standard test functions.
- Show two applications of C-GRASP on real-world problems:
  - Robot kinematics.
  - Chemical equilibrium system.

Experimental Results

- Compare C-GRASP with other global optimization heuristics on a set of standard test functions.
- Show two applications of C-GRASP on real-world problems:
  - Robot kinematics.
  - Chemical equilibrium system.

Experimental Results

- Compare C-GRASP with other global optimization heuristics on a set of standard test functions.
- Show two applications of C-GRASP on real-world problems:
  - Robot kinematics.
  - Chemical equilibrium system.

Experimental Results

- Compare C-GRASP with other global optimization heuristics on a set of standard test functions.
- Show two applications of C-GRASP on real-world problems:
  - Robot kinematics.
  - Chemical equilibrium system.

Experiment Setup

- Experiments run on Dell PowerEdge 2600 computer with dual 3.2 GHz 1 Mb cache XEON III processors and 6 Gb memory running RedHat Linux 3.2.3-53.
- Heuristic implemented in C++ and complied with GNU `g++` version 3.2.3 using options `-O6 -funroll-all-loops -fomit-frame-pointer -march=pentium4`.
- Times measured with `getusage()`.

Experiment Setup

- Experiments run on Dell PowerEdge 2600 computer with dual 3.2 GHz 1 Mb cache XEON III processors and 6 Gb memory running RedHat Linux 3.2.3-53.
- Heuristic implemented in C++ and complied with GNU g++ version 3.2.3 using options `-O6 -funroll-all-loops -fomit-frame-pointer -march=pentium4`.
- Times measured with `getusage()`.

Experiment Setup

- Experiments run on Dell PowerEdge 2600 computer with dual 3.2 GHz 1 Mb cache XEON III processors and 6 Gb memory running RedHat Linux 3.2.3-53.
- Heuristic implemented in C++ and complied with GNU g++ version 3.2.3 using options `-O6 -funroll-all-loops -fomit-frame-pointer -march=pentium4`.
- Times measured with `getusage()`.

C-GRASP Parameters

- C-GRASP has six parameters:
  - RCL parameter: $\alpha = 0.4$
  - Initial search space discretization size: $h = 1$
  - Number of outer loop (multi-start) iterations: $\texttt{NumTimesToRun} = 20$
  - Number of C-GRASP inner iterations: $\texttt{MaxIters} = 200$
  - Maximum number of inner loop iterations without improvement before h is reduced: $\texttt{MaxNumIterNoImprov} = 20$
  - Maximum local improvement directions: $\texttt{MaxDirToTry} = 30$

C-GRASP Parameters

- C-GRASP has six parameters:
  - RCL parameter: $\alpha = 0.4$
  - Initial search space discretization size: $h = 1$
  - Number of outer loop (multi-start) iterations: $\texttt{NumTimesToRun} = 20$
  - Number of C-GRASP inner iterations: $\texttt{MaxIters} = 200$
  - Maximum number of inner loop iterations without improvement before h is reduced: $\texttt{MaxNumIterNoImprov} = 20$
  - Maximum local improvement directions: $\texttt{MaxDirToTry} = 30$

C-GRASP Parameters

- C-GRASP has six parameters:
  - RCL parameter: $\alpha = 0.4$
  - Initial search space discretization size: $h = 1$
  - Number of outer loop (multi-start) iterations: $\texttt{NumTimesToRun} = 20$
  - Number of C-GRASP inner iterations: $\texttt{MaxIters} = 200$
  - Maximum number of inner loop iterations without improvement before h is reduced: $\texttt{MaxNumIterNoImprov} = 20$
  - Maximum local improvement directions: $\texttt{MaxDirToTry} = 30$

C-GRASP Parameters

- C-GRASP has six parameters:
  - RCL parameter: $\alpha = 0.4$
  - Initial search space discretization size: $h = 1$
  - Number of outer loop (multi-start) iterations: $\texttt{NumTimesToRun} = 20$
  - Number of C-GRASP inner iterations: $\texttt{MaxIters} = 200$
  - Maximum number of inner loop iterations without improvement before h is reduced: $\texttt{MaxNumIterNoImprov} = 20$
  - Maximum local improvement directions: $\texttt{MaxDirToTry} = 30$

C-GRASP Parameters

- C-GRASP has six parameters:
  - RCL parameter: $\alpha = 0.4$
  - Initial search space discretization size: $h = 1$
  - Number of outer loop (multi-start) iterations: $\texttt{NumTimesToRun} = 20$
  - Number of C-GRASP inner iterations: $\texttt{MaxIters} = 200$
  - Maximum number of inner loop iterations without improvement before h is reduced: $\texttt{MaxNumIterNoImprov} = 20$
  - Maximum local improvement directions: $\texttt{MaxDirToTry} = 30$

C-GRASP Parameters

- C-GRASP has six parameters:
  - RCL parameter: $\alpha = 0.4$
  - Initial search space discretization size: $h = 1$
  - Number of outer loop (multi-start) iterations: $\texttt{NumTimesToRun} = 20$
  - Number of C-GRASP inner iterations: $\texttt{MaxIters} = 200$
  - Maximum number of inner loop iterations without improvement before h is reduced: $\texttt{MaxNumIterNoImprov} = 20$
  - Maximum local improvement directions: $\texttt{MaxDirToTry} = 30$

C-GRASP Parameters

- C-GRASP has six parameters:
  - RCL parameter: $\alpha = 0.4$
  - Initial search space discretization size: $h = 1$
  - Number of outer loop (multi-start) iterations: $\texttt{NumTimesToRun} = 20$
  - Number of C-GRASP inner iterations: $\texttt{MaxIters} = 200$
  - Maximum number of inner loop iterations without improvement before h is reduced: $\texttt{MaxNumIterNoImprov} = 20$
  - Maximum local improvement directions: $\texttt{MaxDirToTry} = 30$

Other Heuristics

- We compare C-GRASP with other heuristics for global optimization:
  - Enhanced simulated annealing (EAS) of Siarry et al. (1997).
  - Monte Carlo simulated annealing (MCSA) of Vanderbilt and Louie (1984).
  - Sniffer global optimization (SGO) of Butler and Slaminka (1992). Uses gradient information.
  - Directed tabu search (DTS) of Hedar and Fukushima (2006).

Other Heuristics

- We compare C-GRASP with other heuristics for global optimization:
  - Enhanced simulated annealing (EAS) of Siarry et al. (1997).
  - Monte Carlo simulated annealing (MCSA) of Vanderbilt and Louie (1984).
  - Sniffer global optimization (SGO) of Butler and Slaminka (1992). Uses gradient information.
  - Directed tabu search (DTS) of Hedar and Fukushima (2006).

Other Heuristics

- We compare C-GRASP with other heuristics for global optimization:
  - Enhanced simulated annealing (EAS) of Siarry et al. (1997).
  - Monte Carlo simulated annealing (MCSA) of Vanderbilt and Louie (1984).
  - Sniffer global optimization (SGO) of Butler and Slaminka (1992). Uses gradient information.
  - Directed tabu search (DTS) of Hedar and Fukushima (2006).

Other Heuristics

- We compare C-GRASP with other heuristics for global optimization:
  - Enhanced simulated annealing (EAS) of Siarry et al. (1997).
  - Monte Carlo simulated annealing (MCSA) of Vanderbilt and Louie (1984).
  - Sniffer global optimization (SGO) of Butler and Slaminka (1992). Uses gradient information.
  - Directed tabu search (DTS) of Hedar and Fukushima (2006).

Other Heuristics

- We compare C-GRASP with other heuristics for global optimization:
  - Enhanced simulated annealing (EAS) of Siarry et al. (1997).
  - Monte Carlo simulated annealing (MCSA) of Vanderbilt and Louie (1984).
  - Sniffer global optimization (SGO) of Butler and Slaminka (1992). Uses gradient information.
  - Directed tabu search (DTS) of Hedar and Fukushima (2006).

Experiments

- C-GRASP is compared to other heuristics on 14 test problems.
- Global minimum value f^* is known for all problems in test set.
- C-GRASP is run until objective function value f is significantly close to global optimum, i.e. when

$$|f^* - f| \leq 10^{-4}|f^*| + 10^{-6}$$

  or `NumTimesToRun` restarts are done.
- For each problem, 100 independent runs of C-GRASP are done. We record the percentage of runs that find a significantly close solution, the time to find such solutions and the number of function evaluations.

Experiments

- C-GRASP is compared to other heuristics on 14 test problems.
- Global minimum value f^* is known for all problems in test set.
- C-GRASP is run until objective function value f is significantly close to global optimum, i.e. when

$$|f^* - f| \leq 10^{-4}|f^*| + 10^{-6}$$

or `NumTimesToRun` restarts are done.
- For each problem, 100 independent runs of C-GRASP are done. We record the percentage of runs that find a significantly close solution, the time to find such solutions and the number of function evaluations.

Experiments

- C-GRASP is compared to other heuristics on 14 test problems.
- Global minimum value f^* is known for all problems in test set.
- C-GRASP is run until objective function value f is significantly close to global optimum, i.e. when

$$|f^* - f| \leq 10^{-4}|f^*| + 10^{-6}$$

or `NumTimesToRun` restarts are done.

- For each problem, 100 independent runs of C-GRASP are done. We record the percentage of runs that find a significantly close solution, the time to find such solutions and the number of function evaluations.

Experiments

- C-GRASP is compared to other heuristics on 14 test problems.
- Global minimum value f^* is known for all problems in test set.
- C-GRASP is run until objective function value f is significantly close to global optimum, i.e. when

$$|f^* - f| \leq 10^{-4}|f^*| + 10^{-6}$$

  or `NumTimesToRun` restarts are done.
- For each problem, 100 independent runs of C-GRASP are done. We record the percentage of runs that find a significantly close solution, the time to find such solutions and the number of function evaluations.

Experiments

- We use published results for other heuristics.
- ESA and DTS use same measure of closeness as C-GRASP, i.e.

$$|f^* - f| \leq 10^{-4}|f^*| + 10^{-6}.$$

- MCSA and SGO use slightly different measure:

$$|f^* - f| \leq 10^{-3}|f^*|.$$

Experiments

- We use published results for other heuristics.
- ESA and DTS use same measure of closeness as C-GRASP, i.e.

$$|f^* - f| \leq 10^{-4}|f^*| + 10^{-6}.$$

- MCSA and SGO use slightly different measure:

$$|f^* - f| \leq 10^{-3}|f^*|.$$

Experiments

- We use published results for other heuristics.
- ESA and DTS use same measure of closeness as C-GRASP, i.e.

$$|f^* - f| \leq 10^{-4}|f^*| + 10^{-6}.$$

- MCSA and SGO use slightly different measure:

$$|f^* - f| \leq 10^{-3}|f^*|.$$

Branin Function

$$\min \ (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{1}{\pi}5x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_1) + 10$$

$$\text{subject to: } (-5, 10) \leq (x_1, x_2) \leq (0, 15).$$

heuristic	% runs sign. close	func. eval.	avg. time
ESA	-	-	-
MCSA	100	557	-
SGO	100	205	-
DTS	100	212	-
C-GRASP	100	59,857	0.0016s

Easom Function

$$\min \quad -\cos(x_1)\cos(x_2)e^{-(x_1-\pi)^2-(x_2-\pi)^2}$$

$$\text{subject to:} \quad (-100,-100) \leq (x_1,x_2) \leq (100,100).$$

heuristic	% runs sign. close	func. eval.	avg. time
ESA	-	-	-
MCSA	-	-	-
SGO	-	-	-
DTS	82	223	-
C-GRASP	100	89,630	0.0042s

Goldstein-Price Function

$$\min \; [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times$$
$$[30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$$

$$\text{subject to: } (-2, -2) \leq (x_1, x_2) \leq (2, 2).$$

heuristic	% runs sign. close	func. eval.	avg. time
ESA	100	783	-
MCSA	99	1186	-
SGO	100	664	-
DTS	100	230	-
C-GRASP	100	29	0.0000s

Shubert Function

$$\min \ (\sum_{i=1}^{5} i\cos[(i+1)x_1+i])(\sum_{i=1}^{5} i\cos[(i+1)x_2+i])$$

$$\text{subject to: } (-10,-10) \leq (x_1,x_2) \leq (10,10).$$

heuristic	% runs sign. close	func. eval.	avg. time
ESA	-	-	-
MCSA	-	-	-
SGO	-	-	-
DTS	92	274	-
C-GRASP	100	82,363	0.0078s

Hartmann-3 Function

$$\min \quad -\sum_{i=1}^{4} \alpha_i e^{-\sum_{j=1}^{3} A_{ij}^{(3)} (x_j - P_{ij}^{(3)})^2}$$

$$\text{subject to:} \quad (0,0,0) \leq (x_1, x_2, x_3) \leq (1,1,1).$$

heuristic	% runs sign. close	func. eval.	avg. time
ESA	100	698	-
MCSA	100	1224	-
SGO	99	534	-
DTS	100	438	-
C-GRASP	100	20,743	0.0026s

Hartmann-6 Function

$$\min \quad -\sum_{i=1}^{4} \alpha_i e^{-\sum_{j=1}^{6} A_{ij}^{(6)} (x_j - P_{ij}^{(6)})^2}$$

$$\text{subject to: } (0,0,\ldots,0) \leq (x_1, x_2, \ldots, x_6) \leq (1,1,\ldots,1).$$

heuristic	% runs sign. close	func. eval.	avg. time
ESA	100	1638	-
MCSA	62	1914	-
SGO	99	1760	-
DTS	83	1787	-
C-GRASP	100	79,685	0.0140s

Rosenbrock-2 Function

$$\min \quad 100(x_1^2 - x_2)^2 + (x_1 - 1)^2$$

$$\text{subject to:} \quad (-2, -2) \leq (x_1, x_2) \leq (2, 2).$$

heuristic	% runs sign. close	func. eval.	avg. time
ESA	-	-	-
MCSA	-	-	-
SGO	-	-	-
DTS	100	254	-
C-GRASP	100	1,158,350	0.0132s

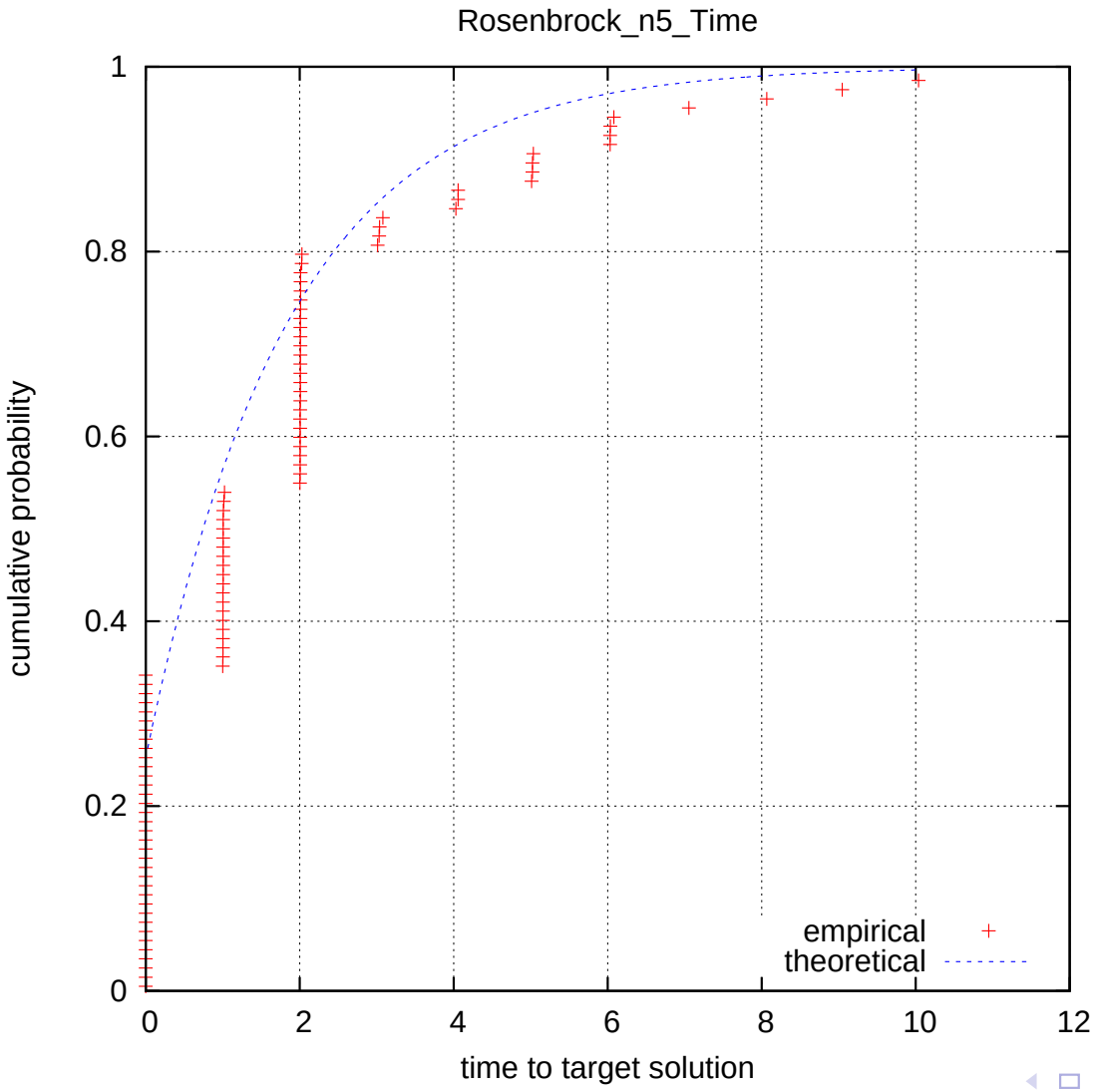
Rosenbrock-5 Function

$$\min \sum_{j=1}^{4} 100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2$$

subject to: $(-2, \ldots, -2) \leq (x_1, \ldots, x_5) \leq (2, \ldots, 2)$.

heuristic	% runs sign. close	func. eval.	avg. time
ESA	-	-	-
MCSA	-	-	-
SGO	-	-	-
DTS	85	1684	-
C-GRASP	100	6,205,503	1.7520s

Rosenbrock-5 Function



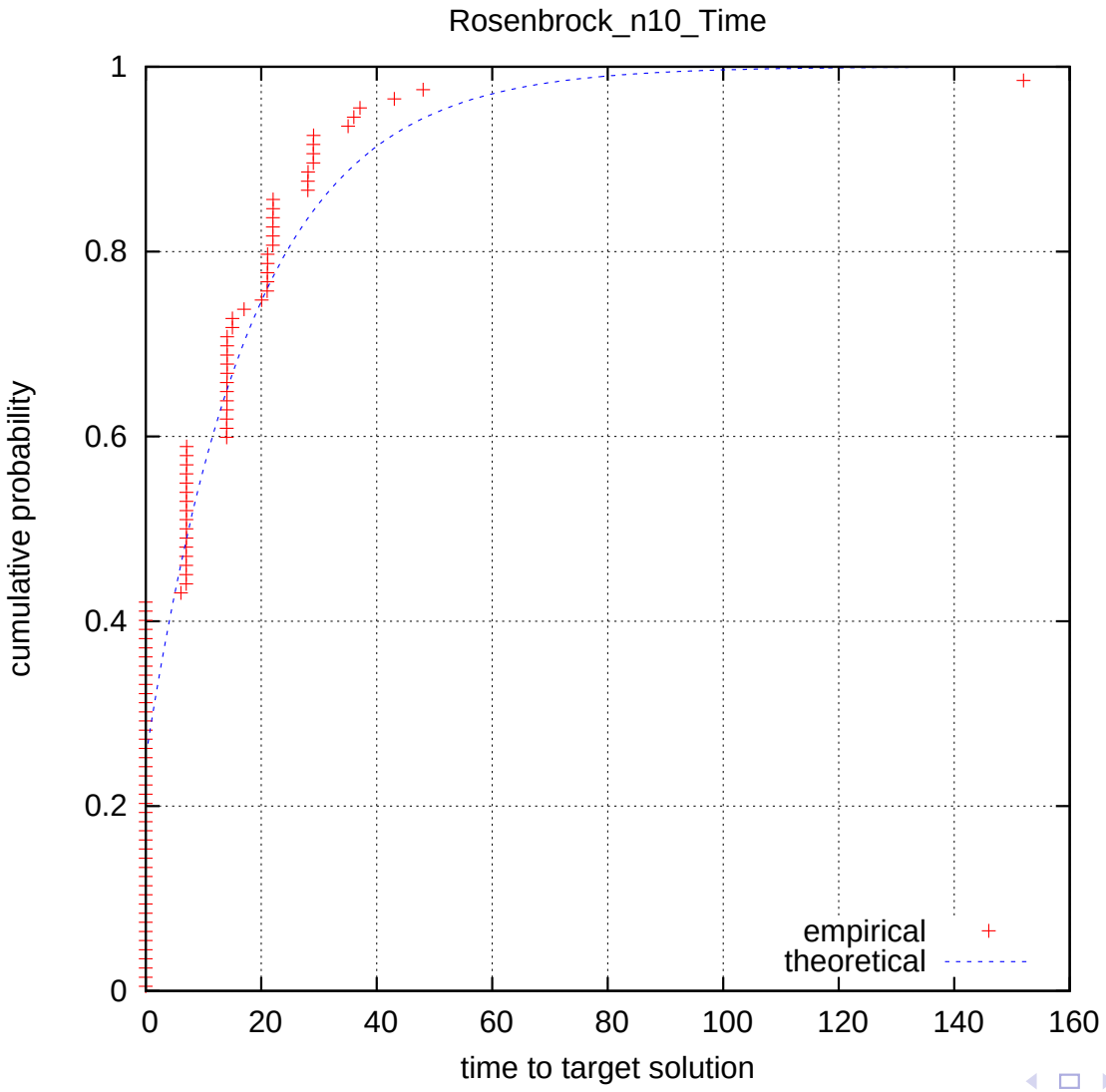
Rosenbrock-10 Function

$$\min \sum_{j=1}^{9} 100(x_j^2 - x_{j+1})^2 + (x_j - 1)^2$$

subject to: $(-2, \ldots, -2) \leq (x_1, \ldots, x_{10}) \leq (2, \ldots, 2).$

heuristic	% runs sign. close	func. eval.	avg. time
ESA	-	-	-
MCSA	-	-	-
SGO	-	-	-
DTS	85	9037	-
C-GRASP	99	20,282,529	11.4388s

Rosenbrock-10 Function



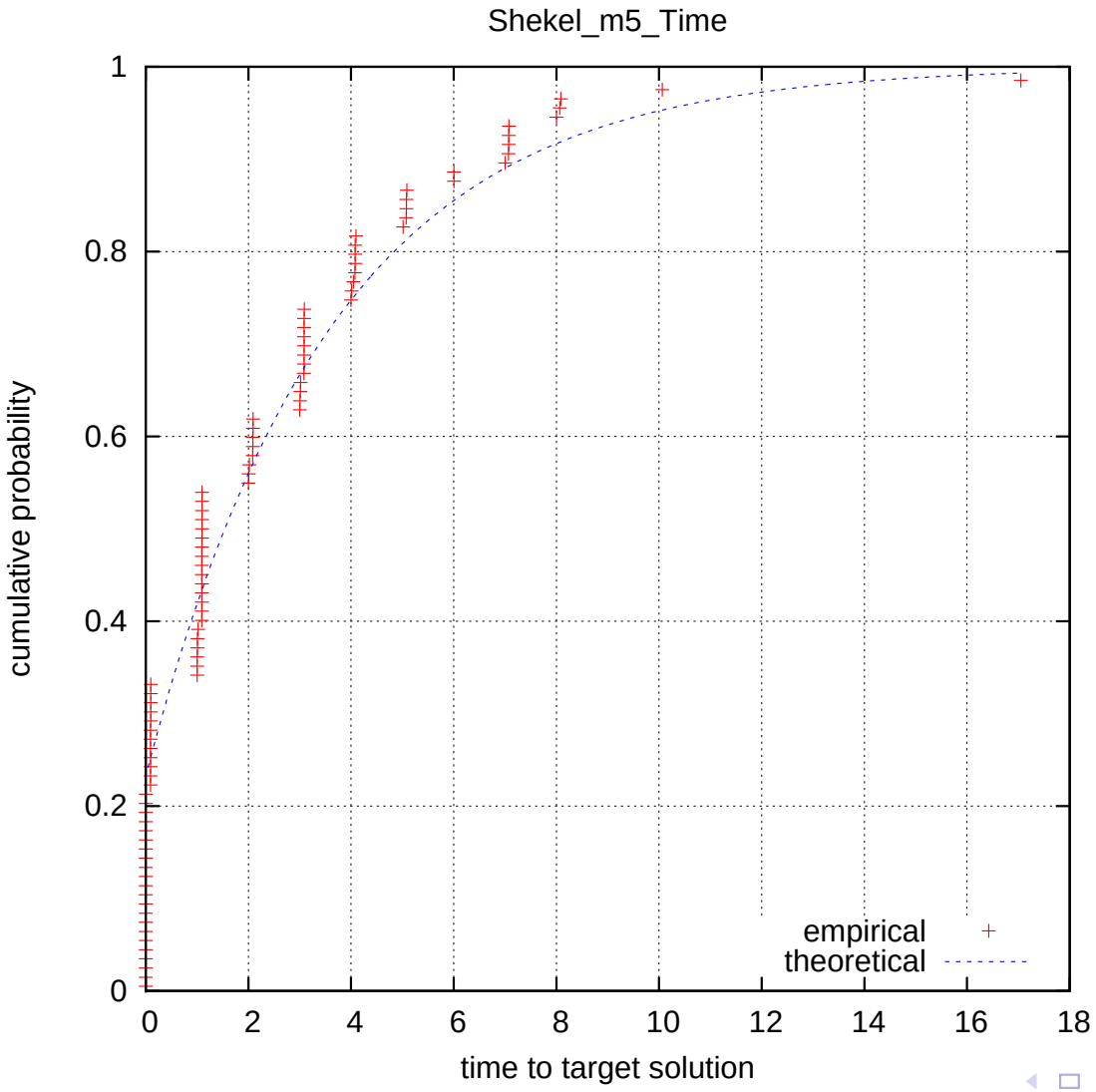
Shekel-(4,5) Function

$$\min \quad -\sum_{i=1}^{5}[(x-\bar{a}_i)^T(x-\bar{a}_i)+c_i]^{-1}$$

subject to: $(0,\ldots,0) \leq (x_1,\ldots,x_5) \leq (10,\ldots,10)$.

heuristic	% runs sign. close	func. eval.	avg. time
ESA	54	1487	-
MCSA	54	3910	-
SGO	90	3695	-
DTS	75	819	-
C-GRASP	100	5,545,982	2.3316s

Shekel-(4,5) Function



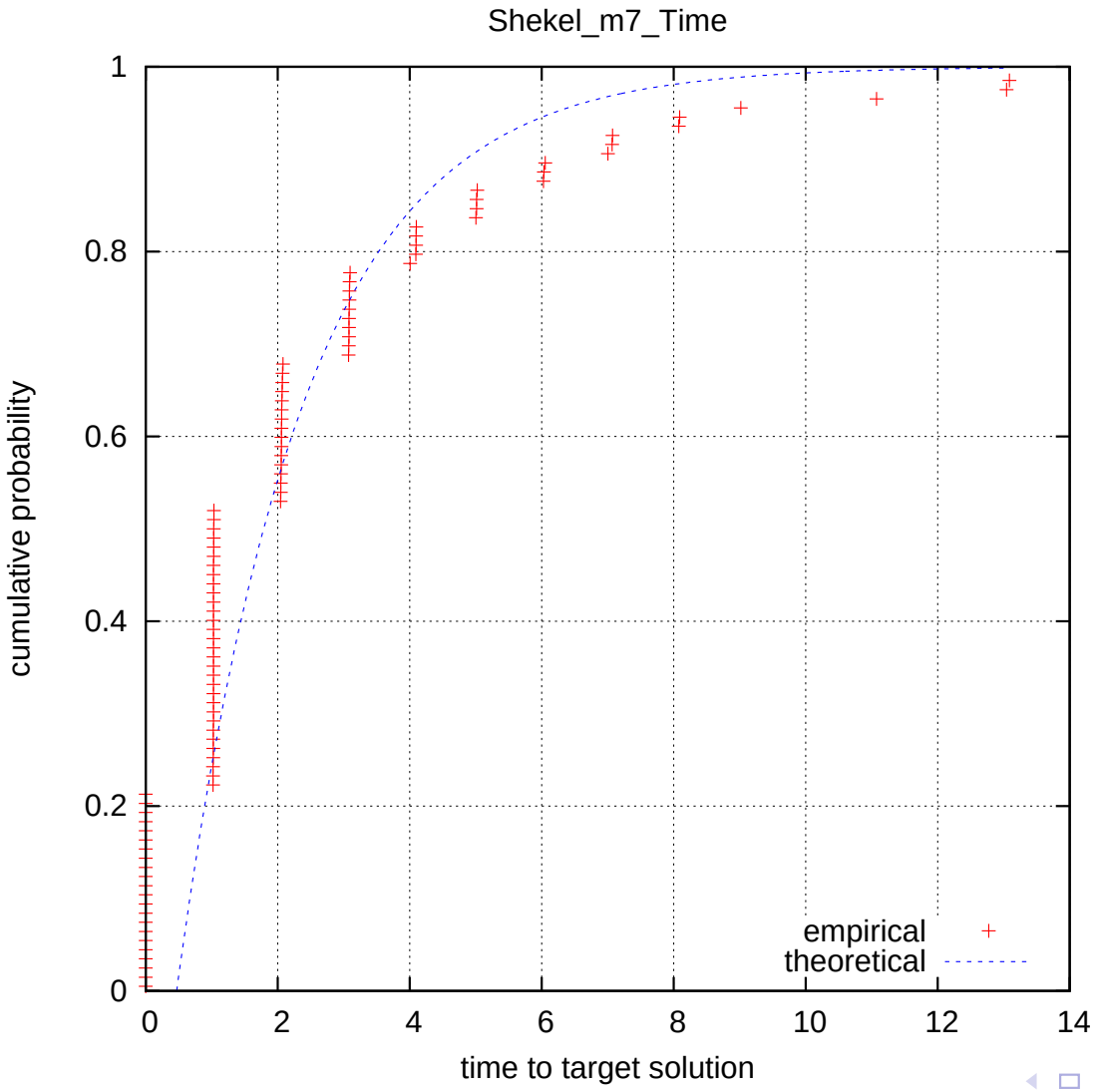
Shekel-(4,7) Function

$$\min \quad -\sum_{i=1}^{7}[(x-\bar{a}_i)^T(x-\bar{a}_i)+c_i]^{-1}$$

subject to: $(0,\ldots,0) \leq (x_1,\ldots,x_7) \leq (10,\ldots,10)$.

heuristic	% runs sign. close	func. eval.	avg. time
ESA	54	1661	-
MCSA	64	3421	-
SGO	96	2655	-
DTS	65	812	-
C-GRASP	100	4,052,800	2.3768s

Shekel-(4,7) Function



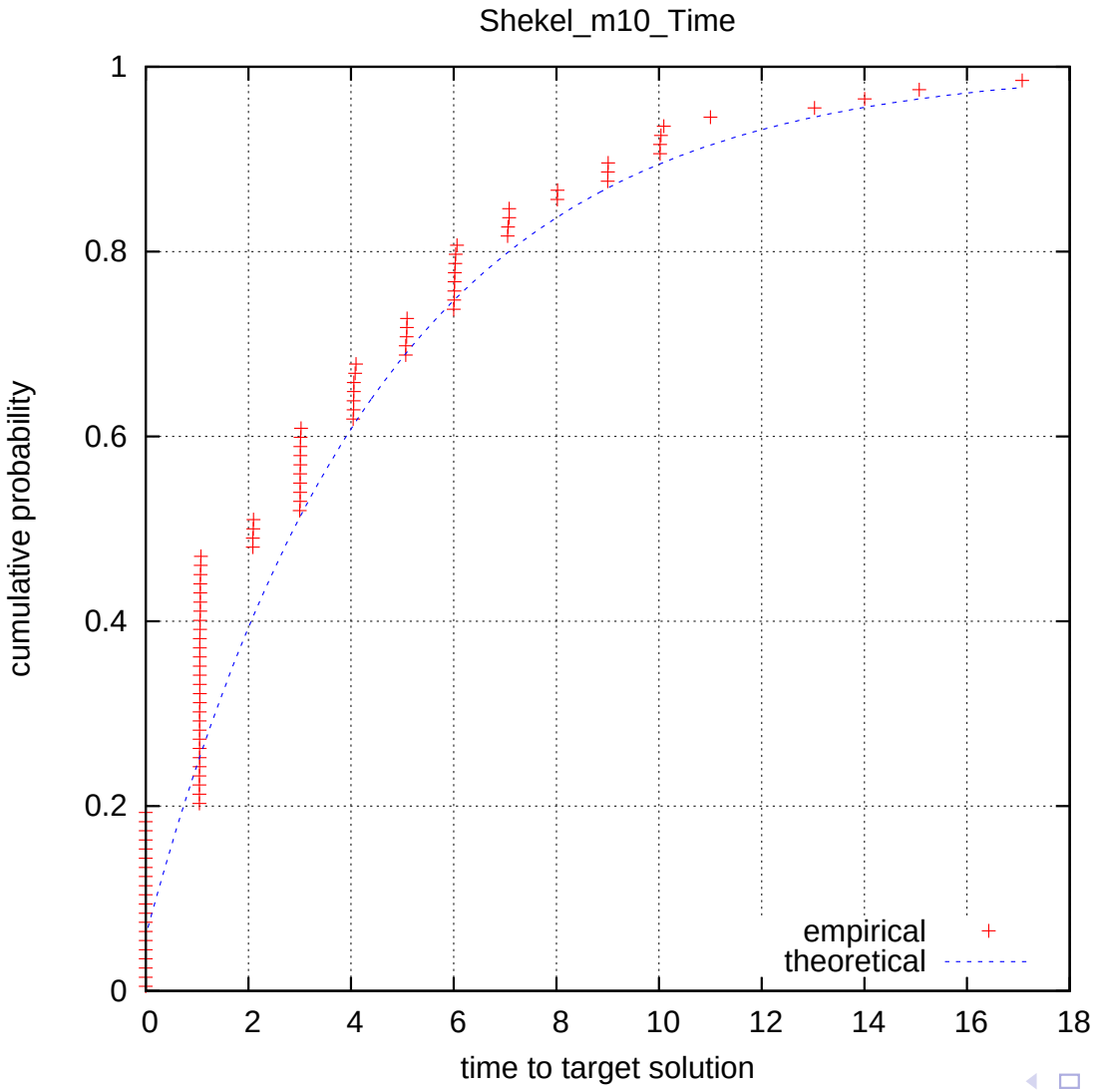
Shekel-(4,10) Function

$$\min \quad -\sum_{i=1}^{10}[(x-\bar{a}_i)^T(x-\bar{a}_i)+c_i]^{-1}$$

$$\text{subject to: } (0,\ldots,0) \leq (x_1,\ldots,x_{10}) \leq (10,\ldots,10).$$

heuristic	% runs sign. close	func. eval.	avg. time
ESA	50	1363	-
MCSA	81	3078	-
SGO	95	3070	-
DTS	52	828	-
C-GRASP	100	4,701,358	3.5172s

Shekel-(4,10) Function



Zakharov-5 Function

$$\min \sum_{i=1}^{5} x_i^2 + \left(\sum_{i=1}^{5} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{5} 0.5 i x_i\right)^4$$

$$\text{subject to: } (-5, \ldots, -5) \leq (x_1, \ldots, x_5) \leq (10, \ldots, 10).$$

heuristic	% runs sign. close	func. eval.	avg. time
ESA	-	-	-
MCSA	-	-	-
SGO	-	-	-
DTS	100	1003	-
C-GRASP	100	959	0.0000s

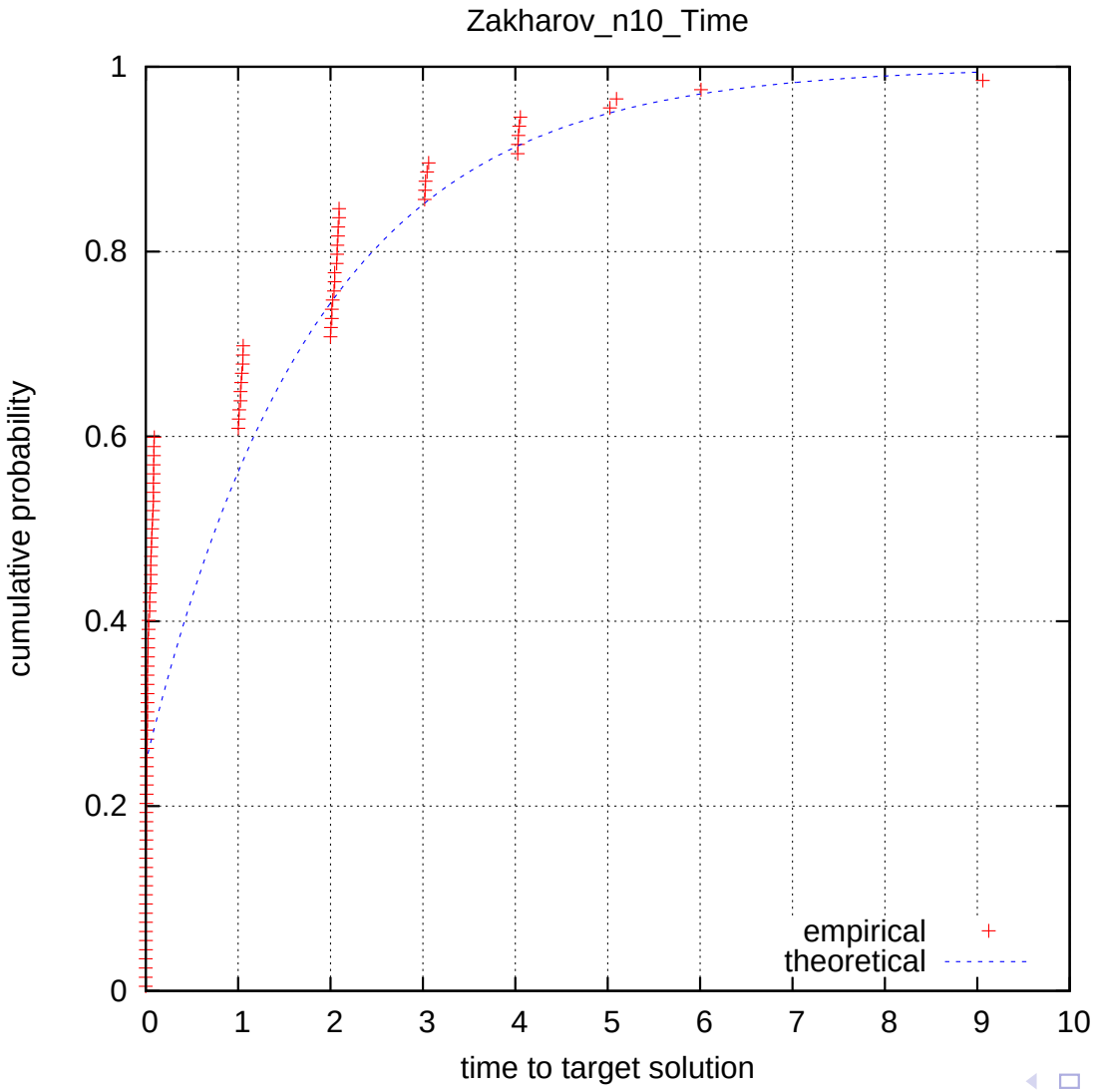
Zakharov-10 Function

$$\min \sum_{i=1}^{10} x_i^2 + \left(\sum_{i=1}^{10} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{10} 0.5 i x_i\right)^4$$

$$\text{subject to: } (-5, \ldots, -5) \leq (x_1, \ldots, x_{10}) \leq (10, \ldots, 10).$$

heuristic	% runs sign. close	func. eval.	avg. time
ESA	-	-	-
MCSA	-	-	-
SGO	-	-	-
DTS	100	4032	-
C-GRASP	100	3,607,653	1.0346s

Zakharov-10 Function



Robot kinematics application

- We consider a robot kinematics application described by Tsai and Morgan (1985).
- Given a 6-revolute manipulator (rigid-bodies, or links, connected together by joints), with the first link designated the base, and the last link designated the hand of the robot: Determine the possible positions of the hand, given that the joints are movable.
- Problem is reduced to solving a system of eight nonlinear equations in eight unknowns.
- Considered a “challenging problem” in Floudas et al. (1999).

Robot kinematics application

- We consider a robot kinematics application described by Tsai and Morgan (1985).
- Given a 6-revolute manipulator (rigid-bodies, or links, connected together by joints), with the first link designated the base, and the last link designated the hand of the robot: Determine the possible positions of the hand, given that the joints are movable.
- Problem is reduced to solving a system of eight nonlinear equations in eight unknowns.
- Considered a "challenging problem" in Floudas et al. (1999).

Robot kinematics application

- We consider a robot kinematics application described by Tsai and Morgan (1985).
- Given a 6-revolute manipulator (rigid-bodies, or links, connected together by joints), with the first link designated the base, and the last link designated the hand of the robot: Determine the possible positions of the hand, given that the joints are movable.
- Problem is reduced to solving a system of eight nonlinear equations in eight unknowns.
- Considered a "challenging problem" in Floudas et al. (1999).

Robot kinematics application

- We consider a robot kinematics application described by Tsai and Morgan (1985).
- Given a 6-revolute manipulator (rigid-bodies, or links, connected together by joints), with the first link designated the base, and the last link designated the hand of the robot: Determine the possible positions of the hand, given that the joints are movable.
- Problem is reduced to solving a system of eight nonlinear equations in eight unknowns.
- Considered a “challenging problem” in Floudas et al. (1999).

Robot kinematics application

Find $x = (x_1, x_2, \ldots, x_8)$ such that:

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

$$f_2(x) = 0.2238 x_1 x_3 + 0.7623 x_2 x_3 + 0.2638 x_1 - x_7 - 0.07745 x_2 - 0.6734 x_4 - 0.6022 = 0$$

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

$$f_4(x) = -0.7623 x_1 + 0.2238 x_2 + 0.3461 = 0$$

$$f_5(x) = x_1^2 + x_2^2 - 1 = 0$$

$$f_6(x) = x_3^2 + x_4^2 - 1 = 0$$

$$f_7(x) = x_5^2 + x_6^2 - 1 = 0$$

$$f_8(x) = x_7^2 + x_8^2 - 1 = 0$$

Robot kinematics application

We form the optimization problem:

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

- Since $F(x) \geq 0$ for all $x \in [-1,1]^8$, then $F(x) = 0 \Longleftrightarrow f_i(x) = 0$ for all $i \in \{1,\ldots,8\}$.
- Hence $\exists\, x^* \in [-1,1]^8 \ni F(x^*) = 0 \Longrightarrow x^*$ is a global minimizer of problem and x^* is a root of the system of equations $f_1(x),\ldots,f_8(x)$.
- There are 16 known roots to this system. Solving problem 16 times using C-GRASP with different starting solutions gives no guarantee of finding all 16 roots.

Robot kinematics application

We form the optimization problem:

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

- Since $F(x) \geq 0$ for all $x \in [-1,1]^8$, then $F(x) = 0 \Longleftrightarrow f_i(x) = 0$ for all $i \in \{1,\ldots,8\}$.
- Hence $\exists\, x^* \in [-1,1]^8 \ni F(x^*) = 0 \Longrightarrow x^*$ is a global minimizer of problem and x^* is a root of the system of equations $f_1(x),\ldots,f_8(x)$.
- There are 16 known roots to this system. Solving problem 16 times using C-GRASP with different starting solutions gives no guarantee of finding all 16 roots.

Robot kinematics application

We form the optimization problem:

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

- Since $F(x) \geq 0$ for all $x \in [-1,1]^8$, then $F(x) = 0 \iff f_i(x) = 0$ for all $i \in \{1,\ldots,8\}$.
- Hence $\exists\, x^* \in [-1,1]^8 \ni F(x^*) = 0 \Longrightarrow x^*$ is a global minimizer of problem and x^* is a root of the system of equations $f_1(x),\ldots,f_8(x)$.
- There are 16 known roots to this system. Solving problem 16 times using C-GRASP with different starting solutions gives no guarantee of finding all 16 roots.

Robot kinematics application

We form the optimization problem:

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

- Since $F(x) \geq 0$ for all $x \in [-1,1]^8$, then $F(x) = 0 \Longleftrightarrow f_i(x) = 0$ for all $i \in \{1,\ldots,8\}$.
- Hence $\exists\, x^* \in [-1,1]^8 \ni F(x^*) = 0 \Longrightarrow x^*$ is a global minimizer of problem and x^* is a root of the system of equations $f_1(x),\ldots,f_8(x)$.
- There are 16 known roots to this system. Solving problem 16 times using C-GRASP with different starting solutions gives no guarantee of finding all 16 roots.

Robot kinematics application

- Suppose the k-th root (roots are denoted $x^1, \ldots, x^k$) has been found.
- Then C-GRASP will restart, with the modified objective function given by:

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

where

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

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.

Robot kinematics application

- Suppose the k-th root (roots are denoted $x^1, \ldots, x^k$) has been found.
- Then C-GRASP will restart, with the modified objective function given by:

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

where

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

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.

Robot kinematics application

- Suppose the k-th root (roots are denoted $x^1, \ldots, x^k$) has been found.
- Then C-GRASP will restart, with the modified objective function given by:

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

where

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

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.

Robot kinematics application

- We made ten independent runs of C-GRASP with $\rho = 1$, $\beta = 10^{10}$, and $\texttt{MaxItersNoImprov} = 5$.
- In each case, the heuristic was able to find all 16 known roots.
- The average CPU time needed to find the 16 roots was 3048 seconds.

Robot kinematics application

- We made ten independent runs of C-GRASP with $\rho = 1$, $\beta = 10^{10}$, and $\texttt{MaxItersNoImprov} = 5$.
- In each case, the heuristic was able to find all 16 known roots.
- The average CPU time needed to find the 16 roots was 3048 seconds.

Robot kinematics application

- We made ten independent runs of C-GRASP with $\rho = 1$, $\beta = 10^{10}$, and $\texttt{MaxItersNoImprov} = 5$.
- In each case, the heuristic was able to find all 16 known roots.
- The average CPU time needed to find the 16 roots was 3048 seconds.

Chemical equilibrium systems

- We examine the chemical reaction that occurs during combustion of propane (C_3H_8) in air (O_2 and N_2).
- Meintjes and Morgan (1990) provide the derivation of this chemical reaction.
- This problem produces a system of ten nonlinear equations in ten unknowns.

Chemical equilibrium systems

- We examine the chemical reaction that occurs during combustion of propane (C_3H_8) in air (O_2 and N_2).
- Meintjes and Morgan (1990) provide the derivation of this chemical reaction.
- This problem produces a system of ten nonlinear equations in ten unknowns.

Chemical equilibrium systems

- We examine the chemical reaction that occurs during combustion of propane (C_3H_8) in air (O_2 and N_2).
- Meintjes and Morgan (1990) provide the derivation of this chemical reaction.
- This problem produces a system of ten nonlinear equations in ten unknowns.

Chemical equilibrium systems

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 (1990) derive a transformation to place the system in canonical form. The canonical form is a system of five nonlinear equations in five unknowns:

$$
\begin{aligned}
g_1 =& y_1 y_2 + y_1 - 3y_5 = 0 \\
g_2 =& 2y_1 y_2 + y_1 + 2R_{10} y_2^2 + y_2 y_3^2 + R_7 y_2 y_3 + \\
& R_9 y_2 y_4 + R_8 y_2 - R y_5 = 0 \\
g_3 =& 2y_2 y_3^2 + R_7 y_2 y_3 + 2R_5 y_3^2 + R_6 y_3 - 8y_5 = 0 \\
g_4 =& R_9 y_2 y_4 + 2y_4^2 + 4R y_5 = 0 \\
g_5 =& y_1 y_2 + y_1 + R_{10} y_2^2 + y_2 y_3^2 + R_7 y_2 y_3 + R_9 y_2 y_4 + \\
& R_8 y_2 + R_5 y_3^2 + R_6 y_3 + y_4^2 - 1 = 0
\end{aligned}
$$

Chemical equilibrium systems

- For both systems, we formed an objective function as the sum of the squares of the nonlinear equations.
- We made ten independent runs of C-GRASP with the parameter `MaxNumIterNoImprov` set to 10.
- For the system in canonical form, C-GRASP was successful on each of the ten runs.
- For the more difficult original system, C-GRASP was successful on eight of the ten runs.

Chemical equilibrium systems

- For both systems, we formed an objective function as the sum of the squares of the nonlinear equations.
- We made ten independent runs of C-GRASP with the parameter `MaxNumIterNoImprov` set to 10.
- For the system in canonical form, C-GRASP was successful on each of the ten runs.
- For the more difficult original system, C-GRASP was successful on eight of the ten runs.

Chemical equilibrium systems

- For both systems, we formed an objective function as the sum of the squares of the nonlinear equations.
- We made ten independent runs of C-GRASP with the parameter `MaxNumIterNoImprov` set to 10.
- For the system in canonical form, C-GRASP was successful on each of the ten runs.
- For the more difficult original system, C-GRASP was successful on eight of the ten runs.

Chemical equilibrium systems

- For both systems, we formed an objective function as the sum of the squares of the nonlinear equations.
- We made ten independent runs of C-GRASP with the parameter `MaxNumIterNoImprov` set to 10.
- For the system in canonical form, C-GRASP was successful on each of the ten runs.
- For the more difficult original system, C-GRASP was successful on eight of the ten runs.

Chemical equilibrium systems

- The average running time for C-GRASP was 37.53 seconds for the canonical system and 201.58 seconds for the original system.
- Meintjes and Morgan (1990) solve the canonical problem by using a variant of 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.

Chemical equilibrium systems

- The average running time for C-GRASP was 37.53 seconds for the canonical system and 201.58 seconds for the original system.
- Meintjes and Morgan (1990) solve the canonical problem by using a variant of 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.

Chemical equilibrium systems

- The average running time for C-GRASP was 37.53 seconds for the canonical system and 201.58 seconds for the original system.
- Meintjes and Morgan (1990) solve the canonical problem by using a variant of 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.

Conclusion

- We describe C-GRASP, a new stochastic local search based metaheuristic for continuous global optimization subject to box constraints that makes no use of gradient information.
- Besides the test problems described in this talk, we have successfully applied C-GRASP to over 150 test problems collected from the literature.
- We have a paper describing the work presented in this talk: M.J. Hirsch, C.N. Meneses, P.M. Pardalos, and M.G.C. Resende, "Global optimization by continuous GRASP," to appear in *Optimization Letters*.
- The paper and these slides can be downloaded from `http://www.research.att.com/~mgcr`.

Conclusion

- We describe C-GRASP, a new stochastic local search based metaheuristic for continuous global optimization subject to box constraints that makes no use of gradient information.
- Besides the test problems described in this talk, we have successfully applied C-GRASP to over 150 test problems collected from the literature.
- We have a paper describing the work presented in this talk: M.J. Hirsch, C.N. Meneses, P.M. Pardalos, and M.G.C. Resende, "Global optimization by continuous GRASP," to appear in *Optimization Letters*.
- The paper and these slides can be downloaded from `http://www.research.att.com/~mgcr`.

Conclusion

- We describe C-GRASP, a new stochastic local search based metaheuristic for continuous global optimization subject to box constraints that makes no use of gradient information.
- Besides the test problems described in this talk, we have successfully applied C-GRASP to over 150 test problems collected from the literature.
- We have a paper describing the work presented in this talk: M.J. Hirsch, C.N. Meneses, P.M. Pardalos, and M.G.C. Resende, "Global optimization by continuous GRASP," to appear in *Optimization Letters*.
- The paper and these slides can be downloaded from `http://www.research.att.com/~mgcr`.

Conclusion

- We describe C-GRASP, a new stochastic local search based metaheuristic for continuous global optimization subject to box constraints that makes no use of gradient information.
- Besides the test problems described in this talk, we have successfully applied C-GRASP to over 150 test problems collected from the literature.
- We have a paper describing the work presented in this talk: M.J. Hirsch, C.N. Meneses, P.M. Pardalos, and M.G.C. Resende, "Global optimization by continuous GRASP," to appear in *Optimization Letters*.
- The paper and these slides can be downloaded from `http://www.research.att.com/~mgcr`.