In: Heuristics: Theory and Applications
Editor: Patrick Siarry, pp. 1-31
ISBN 0000000000

Chapter 1

CONTINUOUS-GRASP REVISITED

Benjamin Martin, Xavier Gandibleux and Laurent Granvilliers *
Université de Nantes — LINA UMR CNRS 6241
UFR Sciences — 2 rue de la Houssinière BP92208, 44322 Nantes Cedex 03 — France

Abstract

C-GRASP is a metaheuristic introduced in 2006 by Hirsch et al. for continuous global optimization. It is a multi-start neighborhood-based metaheuristic derived from the greedy randomized adaptive search procedure proposed by Féo and Resende in 1989. The main difference with most other metaheuristics designed for continuous optimization is the use of a construction procedure. This chapter presents novel mechanisms and components to reinforce the efficiency of the original C-GRASP. The proposals concern (1) the construction procedure, (2) the improvement procedure, and (3) additional new mechanisms. Among the noticeable changes, the improvement procedure is now based on direct searches. The revisions perfect the metaheuristic in reducing its computational effort and facilitating the parameters management. Numerical experiments are performed using benchmark problem functions commonly used in unconstrained continuous global optimization. The collected results are compared with the best results known in the literature for competitive metaheuristics. The good performances of the proposed version confirm the advantage of coupling C-GRASP with direct searches and validate the extensions introduced.

PACS 05.45-a, 52.35.Mw, 96.50.Fm. **Keywords:** Unconstrained Global Optimization, Metaheuristic, GRASP, Direct Search

AMS Subject Classification: 53D, 37C, 65P.

*E-mail address: {firstname}.{lastname}@univ-nantes.fr

Contents

1. Introduction **3**
1.1. Unconstrained global optimization and metaheuristics 3
1.2. Coupling metaheuristics with direct searches 4
1.3. Motivations and directions of our proposals 6

2. From GRASP to C-GRASP **6**
2.1. C-GRASP: versions 2006 and 2010 6
2.2. Construction procedure 8
2.3. Local improvement procedure 11
2.4. Summary of parameters and specificities 11

3. Revisiting C-GRASP: the version 2012 **12**
3.1. Construction procedure with linear complexity 12
3.2. Local improvement by direct search 15
3.3. Control of discretization 15

4. Numerical experiments **16**
4.1. Protocols and benchmarks 17
4.2. Convergence analysis and validation of the propositions 18
4.3. Precision within limited evaluation budget 19
4.4. Strengths and weaknesses 21

5. Conclusion **22**

A Detailed Experimental results **26**
A1. Convergence results 26
A2. Precision within limited evaluation budget 26

B Benchmark Functions **28**

1. Introduction

Dealing with a continuous global optimization problem consists of computing solutions under a set of constraints, and optimizing an objective function. Both constraints and objective function are in general non-linear. Solving such a problem consists of computing an optimal solution. A usual difficulty in this optimization context is the huge number of local optimal solutions. Consequently the computation of a global solution requires a global approach, which is the scope of global optimization. Introduction to global optimization and its applications can be found in [1].

1.1. Unconstrained global optimization and metaheuristics

Unconstrained Global Optimization (UGO) is a special case where the variables are only subject to bound constraints. A UGO problem can be formulated as:

$$\begin{array}{ll} \min & f(x) \\ \text{s.t} & l_i \leq x_i \leq u_i \quad \forall i \in \{1, \cdots, n\} \\ & x \in \mathbb{R}^n \end{array}$$

where $f : D \to \mathbb{R}$ and l_i and u_i are respectively the lower and upper bound of the i^{th} variable, and where $D = \{x \mid l \leq x \leq u\} \subseteq \mathbb{R}^n$. A global optimal solution x^* verifies $\forall x \in [l,u],\ f(x^*) \leq f(x)$. This problem can be seen as the problem of finding the minimum value of a mathematical function. That is why later in this chapter, the term "function" may refer to the related UGO problem. No preliminary hypotheses are made on the functions to solve. Therefore, even if there are no constraints, many difficulties may arise from the function f to minimize. For instance, f may be non-convex, or the gradient of f may not be available, or difficult to use. Moreover the evaluation of f can be time consuming. That is why in some situations, the aim of a solving technique is to give a good solution within a reasonable cost in terms of number of function evaluations. Solving a UGO problem can be done in two main ways:

- **with an exact method**. Here, exact is understood in the sense of the identification of the global optimum x^* by a solution $\hat{x}$ close to it with respect to a fine precision, or the determination of a narrow region $[\hat{l}, \hat{u}]$ containing x^*. The last can be done, for instance, through a branch & bound strategy based on interval analysis. Introduction to these techniques can be found in [2].

- **with an approximation method**. Such methods aim at giving high quality solutions within a restricted computational cost. There is no guarantee of the global optimality of the solutions found. The need of global optimality is not often required in practice. Therefore, this is the most common approach. Metaheuristics illustrate this class of method.

Many metaphors of real life have inspired the research on metaheuristics. Genetic algorithms, tabu search [3], or ant colony [4] are amongst the more representative ones [5,6]. The research on metaheuristics for solving continuous optimization problem has been a very active field for the last few decades. Here the genetic algorithms represent the most important wave of contributions. But recently, variants of metaheuristics originally designed for

combinatorial optimization have been proposed for continuous optimization. The extension for continuous domains of ant colony optimization by Socha and Dorigo [7], or again of tabu search by Chelouah and Siarry [8] are two representative examples. The metaheuristic GRASP (Greedy Randomized Adaptive Search Procedure) [9] has followed this wave with the C-GRASP (for Continuous GRASP) metaheuristic introduced in 2006 by Hirsch et al. [10, 11].

Metaheuristics are often coupled with local search in order to enhance the efficiency of the approximation method [12]. For global optimization, the use of gradient-based techniques with metaheuristics is an interesting option. However, in several situations the gradient is not available. Thus, the coupling of metaheuristics with techniques named "direct searches" appears as an alternative.

1.2. Coupling metaheuristics with direct searches

Direct searches were originally investigated in the 50's - 60's. Their main idea is to extract and use information provided by the evaluations of the problem function, without making any use of its gradient. Although those techniques were widely used due to their understandability and efficiency, direct searches have gained in popularity in continuous optimization field after the work of Torczon in 1991 [13].

Any method which does not use the gradient can be considered as a direct search. However, this definition is wide. In [14], a discussion about this terminology is proposed. In this chapter, we restrain the definition to any method already known as a direct search, or based on one of them. Well known direct searches are the Pattern Search from Hooke and Jeeves [15] and the Simplex Search from Nelder and Mead [16].

The simplex search from Nelder and Mead is known to work well on simple problems (on low-dimensional functions) but can encounter difficulties on other problems. Also, McKinnon shows it has not such good convergence results [17]. Still, its principles are easy to understand, and the method easy to implement and to tune. Moreover, Kelley [18] proposed some tests able to detect degenerated states involved in the convergence issues. When detected, a restarting mechanism is applied. Therefore, two consecutive calls to the simplex search can work as a restarting process. Finally, recent work from Pedroso [19] has introduced simple metaheuristics based on the simplex search. One of them, a multi-start simplex search, shows interesting results compared to its simplicity.

The principles of the original Nelder-Mead from [16] are summarized here. First, from the input solution x, build a simplex of points which is a set of $n+1$ solutions (x + n constructed points). The solutions in the simplex are indexed $x_1, x_2, \cdots, x_{n+1}$ such that $f(x_1) < f(x_2) < \cdots < f(x_{n+1})$. Then, the method tries to improve the worst solution of the simplex x_{n+1} by checking solutions on the line defined by x_{n+1} and the centroid of the other solutions in the simplex $c = \sum_{i=1}^{n} x_i/n$. At least the reflection x_r of x_{n+1} ($x_r = c + (c - x_{n+1})$) of center c is checked and evaluated. Then, under particular conditions, other solutions can be checked and accepted:

- the expansion point $x_e = c + 2(c - x_{n+1})$ is checked if x_r is better than x_1 and accepted if $f(x_e) < f(x_r)$.
- x_r is accepted if $f(x_r) < f(x_n)$.
- the outer contraction $x_{oc} = c + 0.5(c - x_{n+1})$ is checked if $f(x_n) \leq f(x_r) < f(x_{n+1})$ and accepted if $f(x_{oc}) < f(x_r)$.
- the inner contraction $x_{ic} = c - 0.5(c - x_{n+1})$ is checked if $f(x_{n+1}) \leq f(x_r)$ and accepted if $f(x_{ic}) < f(x_{n+1})$.

In the case that none of these solutions is accepted to replace x_{n+1}, a shrinkage of the simplex is made. The simplex is reduced by half towards its best solution. See figure 1 for an illustration of all of those possible trial points in 2 dimensions.

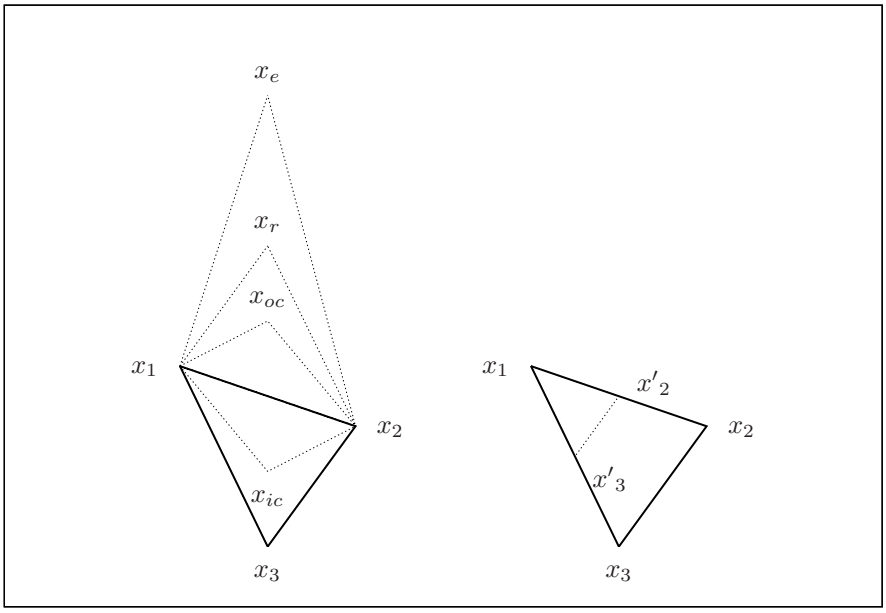

Figure 1. Simplex Search from Nelder and Mead with $n = 2$. On the left, the different possible trial points are presented. On the right, the shrinkage step is shown.

The method is stopped when the difference between the evaluation of the best and the worst solution of the simplex drops below a given threshold or when a number of evaluations is reached.

Recently, direct searches have been successfully combined with metaheuristics. We refer to the PhD thesis of Hedar [20] and the subsequent papers [21–24] for a wide variety of such combinations proposals. We also refer to the recent work from Hvatumm and Glover [25] for an example of a rigorous study of combinations of direct searches with the scatter search metaheuristic in a high-dimensional problem solving context.

1.3. Motivations and directions of our proposals

A targeted aim for us is to solve efficiently unconstrained global optimization problems rigorously with an approach based on interval-based branch & bound algorithm. The pruning of the search domain parts which are proven useless for the determination of the global optimum is a functionality of such an algorithm. The pruning procedure makes use of bounds which are currently heuristically computed. A goal is to improve the quality of bounds by using a fast, aggressive and powerful metaheuristic with, if possible, few parameters to tune. A better bounding gives a better branching process and therefore, it allows faster the elimination of non-promising regions of the search space. Regarding the expected requirements, the overview of the metaheuristics for continuous optimization led us to choose C-GRASP. An in-depth algorithmic study of C-GRASP, and the observations stemming from many numerical experiments convinced us to propose a revisited version of C-GRASP coupled with direct searches.

The next sections of the chapter are organized as follows. Section 2. presents C-GRASP and its main features. The construction and improvement procedures are developed. The parameters are summarized. The revisited version of C-GRASP is presented in Section 3.. The construction procedure with linear complexity in terms of problem function evaluation, the local improvement by direct search and the control of discretization are described. Section 4. states the best algorithmic configuration observed, reports numerical experiments with a comparison with the best metaheuristics from the literature. Section 5. concludes the chapter with a discussion of the propositions and the perspectives of future work around C-GRASP. Two annexes report the extra information related to the numerical experiments.

2. From GRASP to C-GRASP

GRASP is a multi-start, stochastic and neighborhood-based metaheuristic. Its central principle is to mix a greedy algorithm, for the good quality of the solution returned, with a random algorithm, for the diversity of solutions provided. The principle defines the two main phases of GRASP (see Algorithm 1): a greedy randomized *construction phase* (line 1.5), and a local *improvement phase* (line 1.6). For further details on GRASP metaheuristic, we refer to the existing annotated bibliographies (e.g. [26]) and surveys (e.g. [27]) and references therein.

2.1. C-GRASP: versions 2006 and 2010

C-GRASP [10, 11, 28] is an adaptation of GRASP to continuous optimization problems, where the search space is discretized both in the construction and local improvement phases. The construction procedure is a greedy-randomized method which is used to build and/or rebuild a solution. The parameter $\alpha \in [0,1]$ controls the randomness of the construction. A neighborhood is defined and is used inside the local improvement procedure. In addition, the sharpness of both procedures is controlled by a discretization parameter $h \in [h_e, h_s]$ where h_e and h_s are two user defined parameters. They respectively correspond to the ending and starting discretization step.

The main procedure of C-GRASP is presented in Algorithm 2 and explained hereafter.

Algorithm 1: GRASP: High-level algorithm

```
1.1  begin
1.2      InputProblemInstance()
1.3      Initialize(x̂)
1.4      while Stopping criteria not met do
1.5          x ← ConstructGreedyRandomizedSolution(···)
1.6          x′ ← LocalImprovement(x)
1.7          if x′ is better than x̂ then
1.8              x̂ ← UpdateBestSolution(x′)
1.9          end
1.10     end
1.11     return x̂
1.12 end
```

Algorithm 2: C-GRASP: main procedure

```
     Data: The problem (function f, bounds l and u), h_s and h_e, ρ_lo
     Result: The best solution found x̂
2.1  begin
2.2      f̂ ← +∞
2.3      while Stopping criteria not met do
2.4          x ← ConstructUniformRandom(l,u)
2.5          h ← h_s
2.6          while h ≥ h_e do
2.7              ImprC ← false
2.8              ImprL ← false
2.9              (x,ImprC) ← ConstructGreedyRandomized(x,f(.),n,h,l,u) /* Improvement
                 of solution by construction */
2.10             (x,ImprL) ← LocalSearch(x,f(.),n,h,ρ_lo,l,u) /* Improvement of solution by
                 local search */
2.11             if f(x) < f̂ then
2.12                 x̂ ← x
2.13                 f̂ ← f(x)
2.14             end
2.15             if ¬ImprC and ¬ImprL then
2.16                 h ← h/2
2.17             end
2.18         end
2.19     end
2.20     return x̂
2.21 end
```

The first loop at line 2.3 concerns the multi-start part of the method. The next two lines are the initialization of a start. A solution x is randomly created in the search space and the parameter h is set to its initial value h_s. The second loop at line 2.6 is the main loop of the method where the calls to the construction and local improvement procedures occur

respectively at lines 2.9 and 2.10. The construction procedure's inputs are all induced by the problem instance ($f(.)$, n, l and u) and the current solution x. The parameter α is randomly selected in $[0,1]$ at the beginning of each call to the construction procedure. The local improvement has an additional specific input $\rho_{lo} \in [0,1]$ which is a user defined parameter controlling the portion of neighborhood that will be checked. Both procedures have two outputs: a (possibly) modified solution and a boolean value indicating if the procedure has produced an improvement. These values are contained in the variables *ImprC* and *ImprL*. Block starting at line 2.11 updates the best solution found. Finally, line 2.15 is the necessary condition leading to a reduction of the discretization step h. When both the construction and the local improvement procedure do not produce an improvement, the value of h is divided by 2.

Among the differences with the version of GRASP designed for discrete optimization, the output solution of the local improvement procedure is also used as input solution for the construction procedure. A solution x is generated, then the search is performed in a series of "rebuild and improve" stages. Along this process, the discretization step h is reduced, refining the overall search.

The multi-start process is controlled by the stopping rules of Hart [11, 29]. Typically, these rules approximate at the end of each start the probability that the best solution found so far $\hat{x}$ is close to the unknown optimum x^* with respect to a given precision ε. More precisely, the probability $P(f(\hat{x}) \leq f(x^*) - \varepsilon)$ is approximated by $\rho_s(\varepsilon)$ where s is the number of starts already performed. With this approximation, no more starts are applied (and the algorithm stops) when

$$\phi(2\delta\sqrt{s}) - \phi(-2\delta\sqrt{s}) - (1 - \rho_s(\varepsilon))^s \geq 1 - \beta$$

where $\phi(x)$ is the cumulative distribution of the standard normal. δ and β are used to determine the minimum number of starts performed. $1-\beta$ corresponds to the required probability of success and δ to the limit on the difference $P(f(\hat{x}) \leq f(x^*) + \varepsilon) - \rho_s(\varepsilon)$, i.e it determines a number of starts for which the approximated probability can be considered sufficiently close to the exact probability.

2.2. Construction procedure

The main feature of C-GRASP compared to other metaheuristics in continuous optimization is the use of a construction procedure. Algorithm 3 summarizes the main steps of the construction procedure presented in detail in [11]. The idea is, starting from the input x, to check different possible values for each variable. These values can be geometrically represented as hyper-grid points aligned on axis directions starting from x. This hyper-grid is built by discretizing the search space with h. The different values of a variable are checked through the use of a line search procedure. Applied to each variable sequentially, this procedure evaluates the fitness of the different variable values, including the current one and the corresponding boundaries, and returns the best one. Finally, one among the best variables is randomly selected, its associated best value replacing the one in x. The selected variable is then removed from the tested ones until the end of the construction procedure. Therefore, x can move at most one time per axis direction.

Algorithm 3: C-GRASP: Greedy Randomized construction procedure

```
   Data: Solution x, Discretization step h
   Result: The modified solution x, Boolean value ImprC
3.1 begin
3.2   UnFixed ← {1,2,...,n}
3.3   α ← Value ∈ [0,1]
3.4   while UnFixed ≠ ∅ do
3.5     for i ∈ UnFixed do
3.6       t_i ← LineSearch(x,h,i,n,f(.),l,u)
3.7     end
3.8     fmin ← min_{i∈Unfixed}(f(t_i))
3.9     fmax ← max_{i∈UnFixed}(f(t_i))
3.10    RCL ← ∅
3.11    Threshold ← fmin + α*(fmax − fmin)
3.12    for i ∈ UnFixed do
3.13      if f(t_i) ≤ Threshold then
3.14        RCL ← RCL ∪ {i}
3.15      end
3.16    end
3.17    j ← RandomlySelectElement(RCL)
3.18    if f(x) > f(t_j) then
3.19      x ← t_j
3.20      ImprC ← true
3.21    end
3.22    UnFixed ← UnFixed\{j}
3.23  end
3.24  return (x, ImprC)
3.25 end
```

The greedy-randomized selection of a variable works as follows. The application of the line searches along each tested variable computes a set of tested solutions T with $T = \{t_k \mid k \in K\}$ where K is the set of tested directions (or variables) and t_k is the best grid solution along the k^{th} axis direction (initially, $K = \{1..n\}$) starting from the input solution x. The solution t_k can only differ from x on the k^{th} variable value. Then, *fmin* and *fmax* are defined as $fmin = min_{t \in T}(f(t))$ and $fmax = max_{t \in T}(f(t))$. Finally, a *Restricted Candidate List* (*RCL*) is defined as:

$$RCL \;=\; \{t \mid f(t) \leq fmin + \alpha \times (fmax - fmin),\; t \in T\}$$

The solution replacing x will be randomly (uniformly) selected in the *RCL*. The influence of the parameter α can easily be seen: if $\alpha = 0$ then $RCL = \{tmin\}$ where $f(tmin) = fmin$. Thus, the best solution is always selected. Otherwise, if $\alpha = 1$ then $RCL = T$ therefore the next solution is randomly selected between all of the other tested ones. The higher the value of α, the more random the construction procedure. In [11] α is randomly selected at the beginning of each construction procedure, making some calls more random than others. The construction procedure is analogous to standard constructions in discrete optimization. Utility of variables are computed and a selection of one of the variables to change is made among the most interesting ones. In C-GRASP, the utility

used is the evaluation of the function f. Figure 2 shows an example in 2 dimensions of a call to the construction procedure.

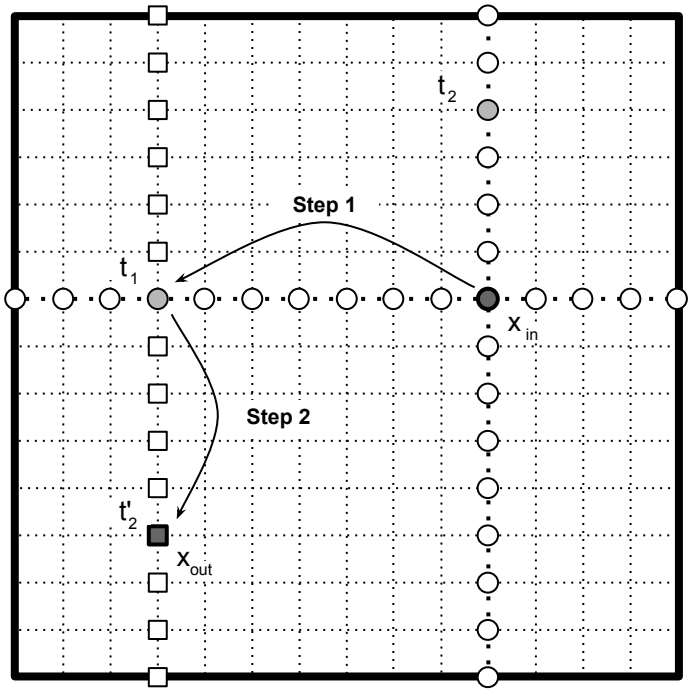

Figure 2. Construction with $n = 2$. The grid is discretized by h. First, compare grid points along each axis and select one of the best between t_1 and t_2 with respect to f. t_1 is chosen thus x is replaced by t_1 at the end of step 1. Then the remaining direction (in squares) is checked and the best trial t_2' is kept. Finally, x is moved to t_2' at the end of step 2, terminating the construction.

The natural translation of the ideas of a construction procedure in combinatorial optimization to non-linear continuous optimization makes the method more dedicated to solve problems having some separability. Since the construction re-builds a solution variable per variable, it is not possible to directly reach a solution with at least 2 different variables value from the input solution. There must be a strictly improving path between the two solutions, on which moves from one solution to another modify at most one variable.

Example 1. *Consider the problem of minimizing* $x_1 \sin(x_1 \cdot \frac{\pi}{2}) + x_2 \sin(x_2 \cdot \frac{\pi}{2})$ *with* $x_1, x_2 \in [1,13]$. *All optima are located on* $\{3,7,11\}^2$ *with the global one on* $(11,11)$. *A call to the construction procedure is made with the input solution* $x = (3,7)$ *and* $h = 4$. *The variables values giving the best improvement starting from those of x are 11 for both variables. The first iteration of the construction will move either to the solution* $(11,7)$ *or* $(3,11)$. *Suppose* $(11,7)$ *is chosen. Then other values have to be checked for the second variable. The best one is again* 11*, leading to the solution* $(11,11)$ *which is the global optimum. In this example, the solution* $x = (3,7)$ *was able to move to* $x^* = (11,11)$ *because each of the intermediate solutions (the solution* $(11,7)$*) improve x.*

2.3. Local improvement procedure

Considering now the local improvement procedure, it appears that we are fairly free to chose the method we want. In C-GRASP [11], the method used is a neighborhood search called *h-neighborhood* search. A set of solutions is built onto the hyper-sphere of radius h centered on the solution to improve x. A solution inside the neighborhood is randomly selected and checked if it improves the current one. In the case that a better solution is found, the new solution replaces the current one x and the process is repeated. Algorithm 4 describes the procedure.

The *h-neighborhood* [11] is defined as follows: let h be the current discretization step. Define

$$S_h(x) = \{x' \mid l \leq x' \leq u,\ x' = x + \tau * h,\ \tau \in \mathbb{Z}^n\}$$

as the set of points in the search space that are on the discrete grid (of step size h) centered on x. Define

$$B_h(x) = \{x' \mid x' = x + h * (\bar{x} - x)/||\bar{x} - x||_2,\ \bar{x} \in S_h(x) \backslash \{x\}\}$$

to be the set of projections of points in $S_h(x)$ onto the hyper-sphere centered on x of radius h. The *h-neighborhood* of the solution x is the set $B_h(x)$. This is where solutions are randomly and uniformly picked. The parameter ρ_{lo} is used here to determine the maximum number of solutions that will be checked without any improvement before stopping the local search. At most a ρ_{lo} portion of the number of grid solutions are checked before assuming there is no more improvement. This maximum number is computed at line 4.2 and 4.3 in Algorithm 4. The selection of a trial solution occurs at line 4.8. The update of the current solution happens in the block at line 4.9.

2.4. Summary of parameters and specificities

The main parameters of C-GRASP the user has to tune are

- starting and ending discretization steps h_s and h_e.
- the density of the neighborhood ρ_{lo} for the local improvement procedure.
- the stopping rules parameters: the precision of the sought solution ε, the required probability β of convergence and the tolerance δ.

Other parameters can be extracted to be tuned by the user, like α used in the construction procedure. Such parameters are self tuned as previously described and in [11].

As a preliminary conclusion, C-GRASP is a metaheuristic based on easily explainable principles, which follow an easily understandable algorithmic description, and require few parameters to tune. C-GRASP is also flexible since the construction and local improvement procedures do not interact directly together. Thus, it is easy to plug ad-hoc procedures according to the optimization problem to solve. For example, Birgin and al. [30] have recently proposed C-GRASP with a gradient-based technique as a local improvement procedure. In our context where the gradient may be unavailable, the hybridization of C-GRASP with direct searches, gradient-free methods known to be efficient, has motivated our work [31]. Such hybridization has already been suggested in [32] but without giving a strong numerical demonstration on the strength of the approach.

Algorithm 4: C-GRASP: *h-neighborhood* search

Data: Solution x, Discretization step h, Grid proportion ρ_{lo}
Result: The modified solution x, Boolean value $ImprL$

4.1 **begin**
4.2 $NumGridPoints \leftarrow \prod_{i=1}^{n} \lceil (u_i - l_i)/h \rceil$
4.3 $NumPointsNoImprove \leftarrow \lceil \rho_{lo} * NumGridPoints \rceil$
4.4 $NumCheckedPoints \leftarrow 0$
4.5 $minF \leftarrow f(x)$
4.6 **while** $NumCheckedPoints < NumPointsNoImprove$ **do**
4.7 $NumCheckedPoints \leftarrow NumCheckedPoints + 1$
4.8 $\bar{x} \leftarrow SelectRandomlyIn(B_h(x))$
4.9 **if** $f(\bar{x}) < minF$ **then**
4.10 $ImprL \leftarrow$ **true**
4.11 $x \leftarrow \bar{x}$
4.12 $minF \leftarrow f(\bar{x})$
4.13 $NumCheckedPoints \leftarrow 0$
4.14 **end**
4.15 **end**
4.16 **return** $(x, ImprL)$
4.17 **end**

Our contributions to C-GRASP are multiple. We have proposed a revised construction procedure which attempts to give a better compromise between cost and efficiency than the original method. We have also studied the possibility of coupling C-GRASP with direct searches, well known derivative-free local optimizers generally used for that kind of problem. Also, we have investigated new components or strategies to use inside C-GRASP aiming at both reducing the cost of the method without deteriorating its performances, and simplifying the tune of some parameters. The next section develops these contributions.

3. Revisiting C-GRASP: the version 2012

The continuous GRASP algorithm does not scale well in general since too many evaluations of the objective function are necessary to reach precise enough solutions. In this section, a construction procedure with linear complexity is introduced and the local improvement method is implemented by direct search. Parameter tuning, in particular for the discretization parameter, is also discussed.

3.1. Construction procedure with linear complexity

The original construction procedure derives a new solution after n successive axis-aligned movements from the current solution. Each step consists of performing a line search for all the remaining dimensions, leading to $O(n^2)$ calls to this procedure. Moreover, every line search explores the whole domain, hence being very expensive when the discretization

parameter decreases. More precisely, the number of solutions to be visited grows exponentially, as shown in the following proposition.

Proposition 1. *The construction procedure makes $O(2^p n^2)$ evaluations of f where p is the number of times the discretization parameter has been reduced.*

Proof. In the worst case, the current solution is modified at each iteration, leading to $0.5n(n+1)$ calls to the line search procedure. Moreover, one line search on the j-th variable requires $(u_j - l_j)/h$ evaluations, and the discretization parameter verifies $h_s = 2^p h$ since it is regularly halved during the algorithm. The stated result directly follows since there are $O(n^2)$ calls to the line search, each one performing $2^p(\max_j(u_j - l_j)/h_s)$ evaluations of f, i.e. $O(2^p)$. □

Our goal is to define a new construction procedure with a linear complexity in order to handle high-dimensional problems. The exponential factor 2^p can be eliminated by considering a window around the current solution which must remain constant during the search. The quadratic factor n^2 can be simplified by performing only one line search per dimension. However, in order to keep an exploration power, one may have to combine the solutions from the samples located on the coordinate axes.

The new algorithm consists of three successive phases. The first phase creates a sampling through an application of one line search per dimension within the window around the current solution x. This window corresponds to the initial domain $[l,u]$ at the beginning of the algorithm and it is halved along with the discretization parameter h. The second phase is a recombination of the sample solutions located on the coordinate axes, aiming at exploring the current window globally. The third and last phase, as in the original continuous GRASP algorithm, is a greedy randomized selection of the output solution.

It turns out that the recombination procedure used in the second phase is a key component. It must be cheap enough to keep a suitable practical complexity, for instance by fixing the number of solutions to be generated. Moreover, it must have the potential of generating good quality solutions everywhere in the window.

More precisely, let X_i be the set of values considered by the line search on the i-th dimension, for $i = 1,\ldots,n$. The ultimate goal is to select the best solution from the Cartesian product $X_1 \times X_2 \times \cdots \times X_n$ such that a few evaluations of f are done. For separable problems, it seems necessary to select the best value from each set X_i (the value leading to a minimum value of f) and to combine those values. For non separable problems, it also seems important to combine sub-optimal values. The spectrum of strategies must be wide. In this work, a very specific technique has been implemented, described as follows. Let $x = (x_1, x_2, \ldots, x_n)$ be the current solution. Suppose that each X_i is a sequence $x'_{i1} < x'_{i2} < \cdots < x'_{ip}$ ordered by increasing values of f and suppose that the X_i share the same size p (otherwise we must consider the size of the smallest set). Finally, let $L_j = \{y^{1,j}, y^{2,j}, \ldots, y^{n,j} : y^{i,j} = (x_1, \ldots, x_{i-1}, x'_{ij}, x_{i+1}, \ldots, x_n)\}$ be the set of axis-aligned solutions of rank j, obtained by the line searches and let M_j the set of solutions of L_j improving x. The set M_j is used to make the j^{th} recombination of solutions. In the case where $|M_j| \leq 1$, M_j is completed by randomly selected (and non-improving) solutions of L_j such

that $|M_j| \geq 2$. The j^{th} recombined solution $u^j = (u_1^j, u_2^j, \ldots, u_n^j)$ will then be defined as

$$u_i^j = \begin{cases} x'_{ij} & \text{if } y^{i,j} \in M_j \\ x_i & \text{otherwise} \end{cases}$$

$|M_j|$ must be greater than 2 in order to recombine an unvisited solution. Note also that if there are more than 2 solutions of L_j improving x, the recombined solution will then exploit perfectly separability property, as in the original construction procedure.

Then, as output of the second phase, the following set of solutions is returned:

$$L_1 \cup \{\hat{u}\}.$$

The lefthand set gathers the best solutions on each axis. The rightmost set consists of the best solution obtained through the recombinations. This combination strategy is depicted in Figure 3. The level lines of a quadratic separable function $f(x_1,x_2) = (x_1-a)^2 + (x_2-b)^2$ appear as dashed lines. The sample solutions are located on the coordinate axes. Some solutions from the recombination technique are drawn, illustrating the capacity to derive good quality solutions. Here, since the problem is separable, the best solution x^* naturally derives from a combination of the best solutions on each axis.

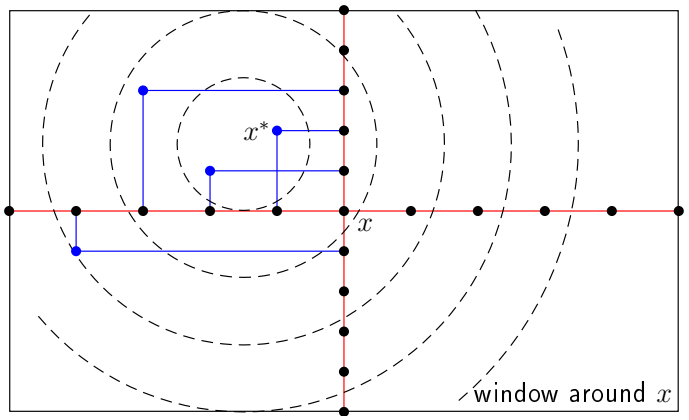

Figure 3. Recombination of sample solutions calculated by line search on axes.

.

The following proposition can now be stated, proving the linear complexity of the new construction procedure.

Proposition 2. *The new construction procedure makes $O(n)$ evaluations of f, provided that the recombination procedure is linear in n.*

Proof. The line search procedure is called n times. The cost of a line search is constant during the search since we consider a window around the current solution whose size decreases along with h. As a consequence, only $\max_j(u_j - l_j)/h_s$ evaluations of f are required, which completes the proof. □

The theoretical complexity is clearly reduced. But the practical efficiency strongly depends on the recombination procedure. During the experimental phase, we will try to show that the new methods reach a balance between the quality of solutions and the number of evaluations of f.

3.2. Local improvement by direct search

The original local improvement procedure is general enough to cope with many categories of problems since it only requires evaluating f. Moreover, after $\lceil \rho_{lo}(\prod_{i=1}^{n} \lceil (u_i - l_i)/h \rceil) \rceil$ evaluations of f, the output solution has a probability ρ_{lo} of being a h-local minimum [11]. Despite good theoretical properties, we have observed that this technique does not reach a good practical efficiency. First, the convergence towards good solutions is not guaranteed in general since the search is unguided and random by nature. Second, the number of evaluations of f grows quickly according to the problem dimension and the reduction of the discretization parameter. Lastly, tuning the density $\rho_{lo} \in [0,1]$ of the neighborhood to be considered may be a hard task.

Our goal is to implement a new derivative-free local search procedure with a better convergence, by exploiting function evaluations to identify descent directions and promising regions. To this end, we propose to implement simplex-based direct search methods. These methods fit well into the continuous GRASP framework for many reasons. First, the discretization parameter can be used to construct simplices around the solution returned by the construction procedure. Second, successive calls act as restarting processes, which is useful to handle degenerated simplices.

We focus here on two simplex-based methods, namely the well-known Nelder-Mead Simplex Search [16] presented in Section 1.2. and the Iterated Simplex Search (ISS) inspired by the work presented in [21]. In the ISS procedure, the simplex is initialized by picking points along coordinate axes from x at distance h. These points are calculated as $x \pm h \cdot e_i$ where e_i is the i-th unit vector, for $i = 1, \ldots, n$. As in NMSS, the first step is an attempt to improve the worst solution x_{n+1}. But in case of failure, this process is iterated on x_n, and so on. When the variable x_j is processed, it suffices to replace x_{n+1} and x_n in NMSS by x_j and x_{j-1}. This iteration replacing the shrinkage step is expected to produce more promising search directions, as illustrated in Figure 4.

The stopping criterion of the ISS procedure is based on three conditions. It stops when a maximum number of evaluations of f is reached, or every vertex of the simplex has been processed, or the difference between the best solution and the worst solution of the simplex drops below a threshold ε_{lo}. In practice, the maximum number of evaluations of f can be automatically assigned to $100n$. This choice may lead to a balance between the two main procedures of the continuous GRASP algorithm (construction and local improvement). Finally, it is worth noticing that parameter ρ_{lo} of the original procedure has been replaced by ε_{lo} representing the precision of the simplex with respect to f. Tuning ε_{lo} seems more natural in the present optimization context.

3.3. Control of discretization

The discretization parameter controls the sharpness of the construction procedure and the local improvement procedure. During one start, this parameter ranges from an initial value h_s to a final value h_e, both being user-defined parameters. In practice, the value of h_s can be automatically assigned to a value involving a reasonable cost of the construction process according to the domain bounds l and u. Tuning h_e is in general more difficult. Indeed a

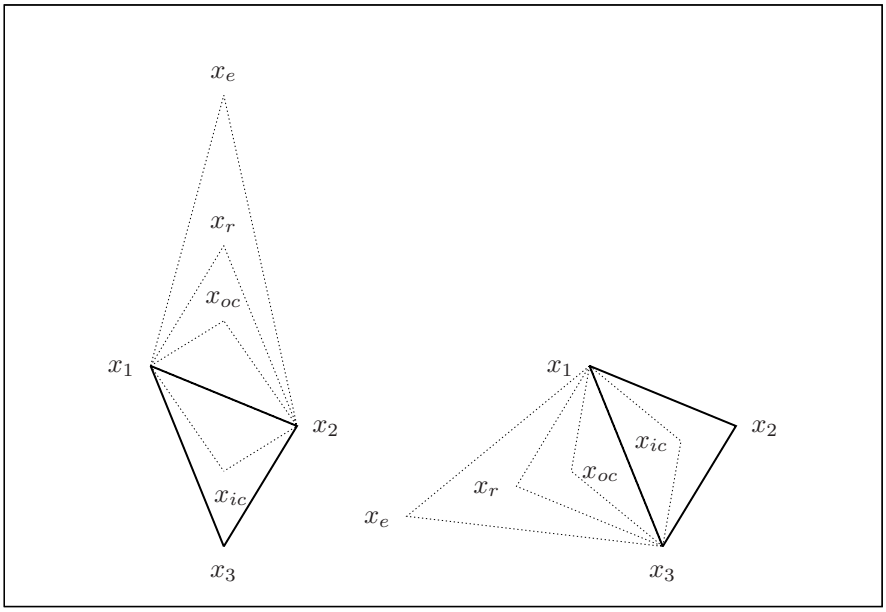

Figure 4. Iterated Simplex Search with $n = 2$. On the left, the first step is to improve x_3. On the right, after a failure on x_3, the process is iterated on x_2.

too small value may lead to slow convergence phenomena and a large value may not lead to reach precise enough solutions.

We propose to manage h_e using an Adaptive Ending Stage (AES). At the beginning of every start, h_e is assigned to h_s. When h becomes strictly smaller than h_e, which arises when the current solution cannot be improved, then an additional step is made. The discretization parameter h is halved and the two procedures are applied in order to improve the current solution x. If the new solution x' significantly improves x, that is $f(x') < f(x) - \varepsilon$, then the search is continued by dividing also h_e by a factor 2. Otherwise the current start is completed. In fact, the setting strategy of parameter h_e is adaptive with respect to the capacity of finding new good solutions. There is no additional parameter since ε is the precision used in the stopping rules.

Another room for improvement consists of limiting the application of the construction procedure. More precisely, for a given value of h, successive constructions are authorized only if each one improves the current solution. Otherwise, either the local improvement procedure is used to improve the current solution or the discretization parameter is decreased to explore the neighborhood more finely. This strategy will be further referred to the Stopping Construction Condition (SCC).

4. Numerical experiments

We implemented our algorithms in C++, using standard double precision computations. Since we will not consider CPU times in the subsequent experiments, results are independent of the configuration of the computer.

4.1. Protocols and benchmarks

The experiments performed are similar those found in some literature, see for instance [8, 11, 21–24, 33]. For each experiment, the *GAP* is defined as

$$GAP = |f(\hat{x}) - f(x^*)|$$

which corresponds to the absolute difference between the best solution found, using a given method, $\hat{x}$ and the known global minimum x^* of the function f.

We propose the following tune of parameters. Given any problem function f and bounds l and u, h_s is set to

$$h_s = 0.05 * min_{i=1}^{n}(u_i - l_i)$$

This way h_s corresponds to 5% of the minimum variable range. The stopping rules parameters δ and β are respectively set to 0.4 and 0.025. This tune allows at least 8 starts to be performed inside a single run on any problem function. The required precision ε will be set relatively to the required precision of each experiment. Note that in any case, we set ε_{lo}, the precision threshold of the simplex-based direct searches, to $0.1 \cdot \varepsilon$.

The set of considered benchmark functions is showed in Table 1 and detailed in Appendix B. The following sub-sections describe different experiments using the *GAP*.

Table 1. Set of benchmark functions. 44 functions whose dimensions vary between 2 and 30.

Function	Dimension	Function	Dimension
Six-Hump Camelback (CA)	2	Beale (BE)	2
Bohachevsky (B_2)	2	Booth (BO)	2
Branin (BR)	2	**Easom** (EA)	2
Goldstein and Price (GP)	2	**Matyas** (M)	2
Rastrigin (RA_2)	2	**Rosenbrock** (R_2)	2
Schwefel (SC_2)	2	**Shubert** (SH)	2
Zakharov (Z_2)	2	**De Jong** (SP_3)	3
Hartmann ($H_{3,4}$)	3	Colville (CV)	4
Perm$_0$ ($P^0_{4,10}$)	4	Perm ($P_{4,\frac{1}{2}}$)	4
Power Sum ($PS_{4,\{8,18,44,114\}}$)	4	**Shekel** ($S_{4,5}$)	4
Shekel ($S_{4,7}$)	4	**Shekel** ($S_{4,10}$)	4
Rastrigin (RA_5)	5	***Rosenbrock*** (R_5)	5
Zakharov (Z_5)	5	**Hartmann** ($H_{6,4}$)	6
Schwefel (SC_6)	6	**Trid** (T_6)	6
Griewank (GR_{10})	10	**Rastrigin** (RA_{10})	10
Rosenbrock (R_{10})	10	Sum Squares (SS_{10})	10
Trid (T_{10})	10	**Zakharov** (Z_{10})	10
Griewank (GR_{20})	20	**Rastrigin** (RA_{20})	20
Rosenbrock (R_{20})	20	Sum Squares (SS_{20})	20
Zakharov (Z_{20})	20	Powell (PW_{24})	24
Dixon and Price (DP_{25})	25	Ackley (A_{30})	30
Levy (L_{30})	30	Sphere (SP_{30})	30

4.2. Convergence analysis and validation of the propositions

Convergence abilities are compared here. As in [8, 11, 24] a convergence condition is defined as

$$GAP \leq 10^{-4} * |f(x^*)| + 10^{-6} \quad (1)$$

It is assumed that a method succeeds in minimizing f if $\hat{x}$, through its GAP, satisfies (1). Any presented methods are applied 100 times to the benchmark consisting of the 24 functions written in boldface in Table 1. Percentage rates PR_s and average evaluations FE_s of runs which successfully converge with respect to (1) are outputted. The required precision ε is set to 10^{-7}.

From the proposals, several implementations of a revised C-GRASP are possible. In order to extract the best one, convergence abilities as described above are compared. To this end, the two measures are merged into a performance measure. The performance measure on a given problem is defined as in [34] as

$$FE_s \cdot \frac{100}{PR_s}$$

Given a set M of methods to compare and a given problem, a method A has performed the best if its performance P_A satisfies $P_A \leq P_B \cdot \tau,\ \forall B \in M \backslash \{A\}$ where τ (set to 1.1) is a smoothing factor allowing a fair comparison of stochastic methods. Therefore two methods can be both considered as the best. Finally, the score of a method A in M is the number of problems of the benchmark for which A is the best with respect to the previous definition.

The set of methods that will be compared are:

- starting from C-GRASP, combine the different proposed main components.
- starting from the previous best method, apply or not the proposed strategies.

This first set of implementations uses either the original or the revised construction procedure and either the Nelder-Mead Simplex Search (NMSS) or the Iterated Simplex Search (ISS). The scores they obtain are reported in Table 2.

Table 2. Comparison of the performances of the different construction and local improvement procedures

Construction	Local Improvement	Score
Original	NMSS	2
Original	ISS	3
Revised	NMSS	13
Revised	ISS	17

It appears clearly that the revised construction procedure outperforms its original implementation. While their performances on 2-dimensional problems are equivalent due to similar costs, the revised version performs better on higher dimensional ones.

The results also highlight the fact that the ISS direct search gives better results than the NMSS. ISS is strictly better on higher dimensional problems. Consequently, the best implementation is the one combining the revised construction procedure and ISS.

The second set of implementations consists of the previous best one using or not the Stopping Construction Condition (SCC) and using or not the Adaptive Ending Stage (AES). Note that h_e does not have to be supplied when using AES. Otherwise, h_e is tuned similarly to h_s as 0.1% of the maximum variable range. The results are reported in Table 3.

Table 3. Comparison of the performances of the use of the different strategies

SCC	AES	Score
No	No	15
No	Yes	14
Yes	No	18
Yes	Yes	19

It follows that the SCC improves the overall performances. To be more precise, it is clearly interesting to use it when considering the problems of 10 or 20 dimensions. Otherwise, SCC leads to similar performances with implementations not using it.

Using AES or not does not seem to radically change the performances. On the one hand, it is useful on problems like the Shekel functions ($S_{4,5}$, $S_{4,7}$ and $S_{4,10}$). On the other hand, it deteriorates the performances on functions like Easom (*EA*), Shubert (*SH*) and Hartmann 3 ($H_{3,4}$). Runs are generally stopped earlier, with the related possible advantages or disadvantages. However, the use of AES makes the tune of C-GRASP easier since the parameter h_e is no more supplied by the user. Therefore, the use of AES is preferred.

We define then the best implementation *ISS-GRASP* as the implementation of C-GRASP using the revised construction procedure, the ISS as local search and the SCC and AES strategies activated. The detailed results of *ISS-GRASP* on this experiment are presented in Appendix A1.. Compared to the original C-GRASP [11], the evaluation costs have been generally widely reduced (of about a ratio 800 on the function Z_{10} for instance) with only small loss of convergence rates.

4.3. Precision within limited evaluation budget

ISS-GRASP is here subjected to a limited evaluation budget. Precision and some general convergence results are compared with those of other metaheuristics. In this experiment an optimality condition is defined as in [11, 24, 33]:

$$GAP \leq \begin{cases} 0.001 \times |f(x^*)| & \text{if } f(x^*) \neq 0 \\ 0.001 & \text{if } f(x^*) = 0 \end{cases} \qquad (2)$$

It is assumed that a problem function f is solved if (2) is satisfied. 100 runs are applied on each function from Table 1, the 40 not written in italic. Results report for each function the average GAP over all the runs after several evaluation steps, and whether (2) is satisfied. The budget is limited to $50,000$ evaluations. Stopping rules are not activated: runs are applied until the budget is reached. Finally, the value of ε is set to 10^{-4}.

The results obtained with *ISS-GRASP* are compared to those of C-GRASP [11], DTS$_{APS}$ [24] and the Scatter Search of Laguna and Marti [33]. For these methods, results are taken from their respective papers. Note that in C-GRASP [11], some results for this experiment may contain inconsistencies. The function Z_{20} is difficult to solve for C-GRASP whereas it manages to give a small GAP early (GAP of 283 after 100 evaluations as reported in [11]). This result is surprising, knowing the high-variability of the function Z_{20}. Thus, it probably contains an inconsistency.

The average GAP over all tested functions, after different evaluation steps, are reported in Table 4. The number of functions solved are shown in Figure 5. In Appendix A2., completely detailed results of *ISS-GRASP* can be found.

Table 4. Mean GAP over all the restrained first benchmarks set.

	Function evaluations						
Methods	100	500	1,000	5,000	10,000	20,000	50,000
Scatter Search	**134.45**	**26.34**	**14.66**	4.96	3.60	3.52	3.46
DTS$_{\text{APS}}$	50,400	43.06	24.26	4.22	1.80	1.70	1.29
C-GRASP	23,610.61	10,185.84	1,341.70	6.20	4.73	3.92	3.02
ISS-GRASP	$> 10^6$	1,949.49	116.629	**1.557**	**0.302**	**0.031**	**0.017**

From Table 4, it appears that *ISS-GRASP* has a lower mean GAP than C-GRASP after 500 evaluations, and a lower than any of the other compared methods after 5,000 evaluations. Individual results on each function show that no problem has a final *GAP* greater than 1. In C-GRASP [11], only the function Z_{20} has a *GAP* greater than 1. For the Scatter Search in [33], the only functions giving a final *GAP* higher than 1 were the function SC_6, RA_{10}, R_{20} and A_{30} with GAP respectively equal to 118.4341, 9.9496, 2.2441 and 5.5033. In [24], no detailed results were given for the DTS$_{APS}$.

Considering now Figure 5, it appears that *ISS-GRASP* solves more problems than the other compared methods after $10,000$ evaluations. Moreover, no other methods are able to solve as many problems at this step, even when the budget is reached. However, *ISS-GRASP* encounters more difficulties early. Since the local-optima are mostly treated by the multi-start process, C-GRASP and *ISS-GRASP* tend to solve fewer functions than the other neighborhood-based metaheuristic DTS_{APS} between 100 and $5,000$ evaluations. C-GRASP is better than *ISS-GRASP* on this interval budget, which is due to the ability of the original construction procedure to solve more low dimensional problems than the revised one.

ISS-GRASP is competitive with respect to the other tested methods. It does not have particular ease of solving some simple problems (difference in terms of functions solved in early stages) but manages to globally perform well on the whole benchmark set even for some more complicated functions (final total of functions solved and mean GAP).

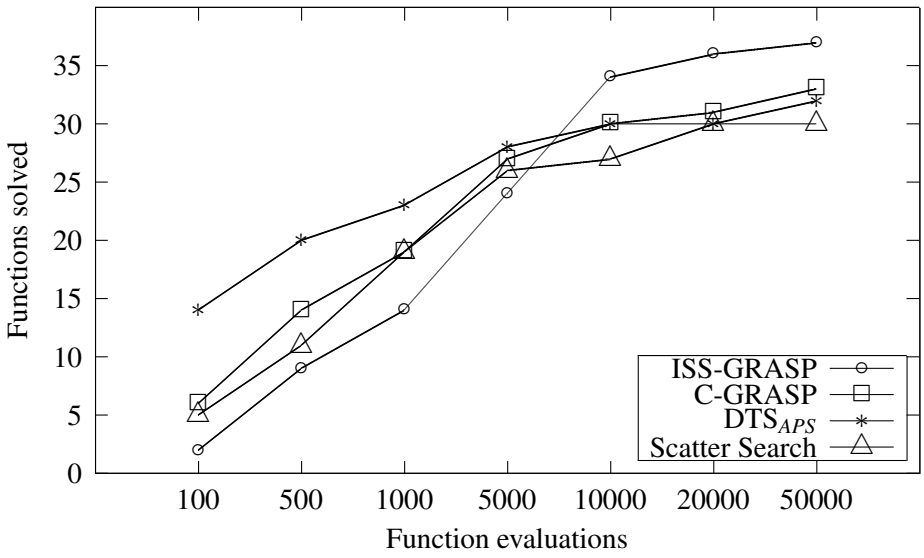

Figure 5. Number of functions solved (GAP verifying (2)) over the restrained first benchmarks set of 40 functions, as a function of function evaluations.

4.4. Strengths and weaknesses

ISS-GRASP is now applied on benchmark functions taken from CEC 2005 [34]. Given the properties of these problems, the performances are studied, allowing us to see the strengths or weaknesses of the proposed approach. To this end, we have selected a few benchmark problems from the CEC 2005 [34]. Table 5 shows a brief description of the properties of the selected functions.

Table 5. Selected problem functions from CEC 2005

Fun	Full Name	Properties
F_2	Shifted Schwefel's Problem 1.2	Unimodal, Non separable
F_3	Shifted Rotated High Conditioned Elliptic Function	Unimodal, Non separable, High variations in fitness
F_4	Shifted Schwefel's Problem 1.2 with Noise	Unimodal, Non separable, Random noise in fitness
F_5	Schwefel's Problem 2.6 with Global Optimum on bounds	Unimodal, Non separable
F_9	Shifted Rastrigin's Function	Multi-modal, Separable
F_{10}	Shifted Rotated Rastrigin's Function	Multi-modal, Non separable
F_{14}	Shifted Rotated Expanded Schaffer's F6	Multi-modal, Non separable

For this experiment, the same protocols as for the preceding experiment in Section 4.3. are used. The difference is that the solving condition is now:

$$GAP \leq 0.001 \quad (3)$$

The dimensions of each function is 10. Results (GAP values) are presented in Table 6.

Table 6. *ISS-GRASP*: GAP values over a selection of problem functions from CEC 2005. Boldface GAP satisfies (3)

	Function evaluations						
Fun	100	500	1,000	5,000	10,000	20,000	50,000
F_2	67732.7	2377.3	857.725	**5.37447e-06**	**4.36103e-06**	**3.73735e-06**	**3.06561e-06**
F_3	1.9311e+09	1.21387e+07	2.84477e+06	31942.9	11846.6	1.23648	**0.000121347**
F_4	83030.7	31644.8	18905.5	2463.95	933.222	305.503	38.5059
F_5	28061.4	4408.22	2999.14	24.9873	0.134177	0.0112214	**0.000889035**
F_9	201.337	24.9026	11.1521	0.00964914	0.00212809	**3.53502e-06**	**3.00037e-06**
F_{10}	291.266	86.218	81.974	49.727	39.0978	30.9627	24.0143
F_{14}	4.63138	4.43081	4.34163	4.08477	3.93758	3.83207	3.68465

ISS-GRASP seems to perform well on F_2, F_3 and F_5. The noisy function F_4 appears to be difficult to solve. Moreover, even F_3 is treated well, high variations in fitness seem to introduce more difficulties in the solving process. Indeed, F_3 is not solved until the budget is reached. In the same way, F_5 is solved by *ISS-GRASP* but with more difficulty. The boundaries may not be well considered.

As attempted, F_9 is solved well. The design of the construction procedure was made to exploit the separability of F_9. But on the other hand, its rotated version F_{10} is not solved. *ISS-GRASP* does not manage to handle non-separability. Finally, it is also not able to solve the complex function F_{14} for similar reasons.

Unimodal or multi-modal and separable functions appears to be the properties exploited the most by *ISS-GRASP*. The introduction of noise and/or high variations in fitness tend to raise difficulties, such as non-separability. The components of *ISS-GRASP* need to be improved to deal with these difficulties.

5. Conclusion

The continuous GRASP metaheuristic is a general algorithmic scheme used to solve unconstrained optimization problems. The exploration of the search space is implemented by a multistart method. Good solutions are obtained by successively applying a construction procedure and a local improvement procedure. The precision of computed solutions is controlled by a discretization step. Specific instances of this framework have been shown to compete well with other metaheuristics. But it appears that these techniques do not scale well in general, requiring a huge number of function evaluations to reach precise solutions.

Our first motivation was to understand precisely the capacities of C-GRASP, and in particular the advantage of the construction procedure. Our second motivation, based on a precise analysis of existing algorithms, was to implement new algorithms as instances of the general scheme. To this end, we have introduced a new construction procedure exploring a restricted neighborhood of the current solution to construct a good quality solution. As a consequence, the global exploration of the search space is left to the multistart method and only a reasonable part of this space is considered in every run.

Second, we have implemented the local improvement procedure by direct searches. We have shown that the iterated simplex search scales better than the Nelder-Mead simplex

search. Combining the revised construction procedure and the iterated simplex search leads to a new instance of C-GRASP that is very competitive. In particular, this new algorithm outperforms the original C-GRASP algorithm and the direct tabu search method.

Third, we have proposed to control automatically the discretization step according to a given threshold on the number of function evaluations that the components will cost. The discretization step is initially assigned to a value proportional to the threshold and it is decreased until no significant improvement of the current solution can be obtained. In so doing, two user-defined parameters are no longer required.

Several interesting directions for future research can be identified. Within the construction procedure, the neighborhood is actually built as an axis-aligned grid of points, but other spanning directions may be more efficient for non separable problems. This can be done for instance by using variable correlation handling techniques as in [35]. Moreover, it may be possible to use another utility function than the objective function, which is very expensive. Local improvement procedures may be selected according to problem characteristics (separability, partial separability, dimension). Finally, good management strategies for the multistart process may lead to being able to identify promising regions of the search space.

References

[1] R. Horst, P. Pardalos, and N. Thoai, *Introduction to global optimization*. Nonconvex optimization and its applications, Kluwer Academic Publishers, 1995.

[2] E. Hansen and G. Walster, *Global optimization using interval analysis*. Pure and applied mathematics, Marcel Dekker, 2004.

[3] F. Glover, "Future paths for integer programming and links to artificial intelligence," *"Computers & Operations Research"*, vol. 13, pp. 533–549, May 1986.

[4] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed Optimization by Ant Colonies," in *First European Conference on Artificial Life (ECAL), Paris, France*, pp. 134–142, 1991.

[5] C. Reeves, *Modern Heuristic Techniques for Combinatorial Problems.* Advanced Topics in Computer Science, McGrawHill, London, 1995.

[6] F. Glover and G. Kochenberger, eds., *Handbook of Metaheuristics*. Kluwer academic publishers, 2002.

[7] K. Socha and M. Dorigo, "Ant colony optimization for continuous domains," *European Journal of Operational Research*, vol. 185, pp. 1155–1173, 2008.

[8] R. Chelouah and P. Siarry, "Tabu search applied to global optimization," *European Journal of Operational Research*, vol. 123, pp. 256–270, 2000.

[9] T. Feo and M. Resende, "Greedy randomized adaptive search procedures," *Journal of Global Optimization*, vol. 6, pp. 109–134, 1995.

[10] M. Hirsch, C. Meneses, P. Pardalos, and M. Resende, "Global optimization by continuous GRASP," *Optimization Letters*, vol. 1, no. 2, pp. 201–212, 2006.

[11] M. Hirsch, M. Resende, and P. Pardalos, "Speeding up continuous GRASP," *European Journal of Operational Research*, vol. 205, no. 3, pp. 507–521, 2010.

[12] C. Blum, M. B. Aguilera, A. Roli, and M. Sampels, eds., *Hybrid Metaheuristics. An Emerging Approach to Optimization*, vol. 114 of *Studies in Computational Intelligence*. Springer, 2008.

[13] V. Torczon, "On the Convergence of the Multidirectional Search Algorithm," *SIAM Journal on Optimization*, vol. 1, pp. 123–145, Feb. 1991.

[14] T. G. Kolda, R. M. Lewis, and V. Torczon, "Optimization by direct search: New perspectives on some classical and modern methods," *SIAM Review*, vol. 45, no. 3, pp. pp. 385–482, 2003.

[15] R. Hooke and T. Jeeves, "Direct search solution of numerical and statistical problems," *Journal of the ACM*, vol. 8, pp. 212–221, 1961.

[16] J. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[17] K. McKinnon, "Convergence of the Nelder-Mead simplex method to a nonstationary point," *SIAM Journal on Optimization*, vol. 9, no. 1, pp. 148–158, 1998.

[18] C. Kelley, "Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease conditions," *SIAM Journal on Optimization*, vol. 10, pp. 43–55, 1999.

[19] J. Pedroso, "Simple metaheuristics using the simplex algorithm for non-linear programming," in *SLS'07 Proceedings of the 2007 international conference on Engineering stochastic local search algorithms: designing, implementing and analyzing effective heuristics*, pp. 217–221, 2007.

[20] A. Hedar, *Studies on metaheuristics for continuous global optimization*. PhD thesis, Kyoto University, 2004.

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

[22] A. Hedar and M. Fukushima, "Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization," *Optimization Methods and Software*, vol. 19, pp. 291–308, 2004.

[23] A. Hedar and M. Fukushima, "Minimizing multimodal functions by simplex coding genetic algorithm," *Optimization Methods and Software*, vol. 18, pp. 265–282, 2003.

[24] A. Hedar and M. Fukushima, "Tabu search directed by direct search methods for nonlinear global optimization," *European Journal of Operational Research*, vol. 170, pp. 329–349, 2003.

[25] L. M. Hvattum and F. Glover, "Finding local optima of high-dimensional functions using direct search methods," *European Journal of Operational Research*, vol. 195, no. 1, pp. 31 – 45, 2009.

[26] P. Festa and M. Resende, "GRASP : an annotated bibliography," in *Essays and surveys on metaheuristics* (C. Ribeiro and P. Hansen, eds.), pp. 325–367, Kluwer academic publishers, 2002.

[27] M. Resende and C. Ribeiro, "Greedy randomized adaptive search procedures," in *Handbook in Metaheuristics* (F. Glover and G. Kochenberger, eds.), pp. 219–249, Kluwer academic publishers, Boston, 2002.

[28] M. Hirsch, *GRASP-based heuristics for continuous global optimization problems*. PhD thesis, University of Florida, 2006.

[29] W. E. Hart, "Sequential stopping rules for random optimization methods with applications to multistart local search," *SIAM Journal on Optimization*, vol. 9, pp. 270–290, May 1998.

[30] E. Birgin, E. Gozzi, M. Resende, and R. Silva, "Continuous GRASP with a local active-set method for bound-constrained global optimization," *Journal of Global Optimization*, vol. 48, pp. 289–310, 2010.

[31] B. Martin, X. Gandibleux, and L. Granvilliers, "Coupling C-GRASP with direct search methods." EVOLVE 2011, May 25-27, Luxembourg.

[32] T. M. Ugulino de Araújo, L. dos Anjos Formiga Cabral, and R. Quirino do Nascimento, "Hybridizing C-GRASP metaheuristics using the adaptive pattern search method to solve global continuous optimization problems," *GEPROS. Gestão da Produção, Operações e Sistemas.*, vol. 4, no. 4, pp. 155–167, 2008. (in portuguese).

[33] M. Laguna and R. Marti, "Experimental testing of advanced scatter search designs for global optimization of multimodal functions," *Journal of Global Optimization*, vol. 33, pp. 235–255, 2005.

[34] P. Suganthan, N. Hansen, J. Liang, K. Deb, Y. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization.," tech. rep., Nanyang Technological University, Singapore, 2005. http://www.ntu.edu.sg/home/epnsugan/.

[35] A. Karimi, H. Nobahari, and P. Siarry, "Continuous ant colony system and tabu search algorithms hybridized for global minimization of continuous multi-minima functions," *Computational Optimization and Applications*, vol. 45, pp. 639–661, 2010.

A Detailed Experimental results

A1. Convergence results

The results presented here refer to the experiment described in Section 4.2..

Table 7. Convergence rates (with respect to (1)) and average number of evaluations of successful runs for *ISS-GRASP*.

Fun	Mean Eval.	% Successful convergences	Fun	Mean Eval.	% Successful convergences
BR	98	100 %	R_2	214	100 %
EA	954	82 %	R_5	1,052	100 %
GP	186	100 %	R_{10}	4,387	100 %
M	98	100 %	R_{20}	20,082	100 %
SH	561	98 %			
SP_3	147	100 %	Z_2	105	100 %
T_6	360	100 %	Z_5	415	100 %
			Z_{10}	2,600	100 %
$H_{3,4}$	275	100 %	Z_{20}	13,444	100 %
$H_{6,4}$	869	100 %			
			RA_2	324	100 %
$S_{4,5}$	1,814	99 %	RA_5	1,170	100 %
$S_{4,7}$	1,581	100 %	RA_{10}	3,014	100 %
$S_{4,10}$	1,948	95 %	RA_{20}	8,636	100 %

A2. Precision within limited evaluation budget

The results presented here refer to the experiment described in Section 4.3..

Table 8. *ISS-GRASP*: GAP values over the first set of benchmark functions.

	Function evaluations						
Fun	100	500	1,000	5,000	10,000	20,000	50,000
CA	0.185773	**1.57054e-06**	**9.48634e-07**	**3.74382e-07**	**2.42487e-07**	**1.39626e-07**	**8.09467e-08**
BE	0.379823	0.0813224	0.0358601	**9.80255e-07**	**7.84168e-07**	**6.35646e-07**	**4.87646e-07**
B_2	0.117019	0.0250845	0.012537	0.00150878	**0.000255162**	**3.45423e-05**	**1.32177e-06**
BO	**8.37733e-05**	**2.08549e-06**	**1.68781e-06**	**1.02332e-06**	**8.04433e-07**	**6.2709e-07**	**5.05435e-07**
BR	0.069267	**2.30108e-06**	**1.84896e-06**	**1.26605e-06**	**1.09113e-06**	**9.82816e-07**	**8.54016e-07**
EA	0.97975	0.577119	0.353667	0.0146875	**1.38341e-06**	**9.82156e-07**	**6.70088e-07**
GP	15.3345	1.53719	**1.88048e-06**	**9.08412e-07**	**6.53757e-07**	**4.81848e-07**	**2.82471e-07**
M	**1.09676e-05**	**1.94601e-06**	**1.5657e-06**	**8.90856e-07**	**7.1863e-07**	**5.69038e-07**	**4.28954e-07**
R_2	2.07764	**2.81362e-06**	**1.73697e-06**	**8.81186e-07**	**6.10907e-07**	**4.74131e-07**	**3.18117e-07**
SC_2	7.372	**2.77248e-05**	**2.6968e-05**	**2.63683e-05**	**2.62144e-05**	**2.60769e-05**	**2.59429e-05**
SH	39.8711	5.35356	1.25289	**7.62838e-06**	**8.00512e-06**	**8.22791e-06**	**8.3716e-06**
Z_2	0.00362545	**1.99456e-06**	**1.58807e-06**	**8.94958e-07**	**7.17312e-07**	**5.72512e-07**	**4.59257e-07**
SP_3	0.00782038	**2.89404e-06**	**2.1434e-06**	**1.34769e-06**	**1.08268e-06**	**8.29054e-07**	**6.08796e-07**
$H_{3,4}$	0.175158	**0.00182942**	**0.000864644**	**1.46785e-06**	**1.26756e-06**	**9.89441e-07**	**7.03763e-07**
CV	1437.79	0.651354	**0.000858366**	**1.99895e-06**	**1.44901e-06**	**1.19114e-06**	**8.5798e-07**
$P^0_{4,10}$	316.846	0.00413423	0.00284131	**4.43176e-05**	**1.01892e-05**	**3.86421e-06**	**2.00096e-06**
$P_{4,\frac{1}{2}}$	2175	0.540259	0.0618133	0.00154182	**0.000639107**	**0.0001803**	**2.58044e-05**
$PS_{4,b}$	35.2065	0.0356309	**0.000420719**	**1.41257e-05**	**3.98483e-06**	**1.85488e-06**	**1.1038e-06**
$S_{4,5}$	9.1082	4.03129	2.54109	0.101506	**2.24384e-06**	**2.68186e-06**	**3.09475e-06**
$S_{4,7}$	9.23534	4.62249	2.45257	0.159317	**0.00011981**	**0.000120244**	**0.000120724**
$S_{4,10}$	9.26047	5.23713	3.62238	0.160939	**0.000124304**	**0.000124719**	**0.000125124**
$H_{6,4}$	2.45182	0.0398398	0.028357	**4.99458e-06**	**4.44917e-06**	**4.03723e-06**	**3.58457e-06**
SC_6	1898.7	22.1066	**8.47781e-05**	**7.92681e-05**	**7.87971e-05**	**7.83904e-05**	**7.78663e-05**
T_6	1202.63	0.166364	**4.48182e-06**	**2.77842e-06**	**2.23573e-06**	**1.9109e-06**	**1.57594e-06**
GR_{10}	173.255	0.962445	0.777594	0.296868	0.132332	0.0402642	0.0148586
RA_{10}	136.673	5.76724	5.51632	**4.89433e-06**	**4.12264e-06**	**3.57647e-06**	**2.9504e-06**
R_{10}	1.22742e+06	252.545	59.3774	0.251605	**4.7749e-06**	**3.73426e-06**	**3.24263e-06**
SS_{10}	1115.66	0.300549	0.0349534	**4.41121e-06**	**3.81212e-06**	**3.30955e-06**	**2.76927e-06**
T_{10}	22503.5	237.237	40.4839	**4.56015e-06**	**3.94154e-06**	**3.45142e-06**	**2.92736e-06**
Z_{10}	4.28964e+07	37.6697	18.3621	**5.33713e-06**	**4.36609e-06**	**3.79391e-06**	**3.07129e-06**
GR_{20}	396.786	1.177	0.864204	0.453936	0.21969	0.0542097	0.00165853
RA_{20}	308.97	13.7855	10.9767	2.84514	**6.87725e-06**	**5.99077e-06**	**5.06253e-06**
R_{20}	3.17289e+06	14263	1659.79	38.8716	10.592	0.479856	**7.33608e-06**
SS_{20}	4662.72	3.39954	2.25868	**1.02377e-05**	**6.0909e-06**	**5.42215e-06**	**4.74094e-06**
Z_{20}	8.43181e+09	37554.2	200.016	16.8748	0.383092	**1.16342e-05**	**8.51746e-06**
PW_{24}	35033.7	4773.75	221.03	0.0625973	**2.73953e-05**	**1.2579e-05**	**7.34982e-06**
DP_{25}	2.57337e+06	20255.4	2426.98	1.27028	0.680295	0.656451	0.646672
A_{30}	20.438	20.3577	7.98171	0.872895	0.0550456	**2.24355e-05**	**1.91957e-05**
L_{30}	362.445	341.498	0.284325	0.0548048	**9.03922e-06**	**7.74158e-06**	**6.57396e-06**
SP_{30}	181.276	173.574	0.0430663	**3.58283e-05**	**7.80805e-06**	**6.84126e-06**	**5.95088e-06**

Table 9. *ISS-GRASP*: Average and standard deviation GAP, and number of functions solved (with respect to (2)) on the first set of benchmark functions.

	Function evaluations						
Measures	100	500	1,000	5,000	10,000	20,000	50,000
Mean GAP	2.12044e+08	1949.49	116.629	1.55736	0.301596	0.0307867	0.0165918
Std. Dev. GAP	1.31623e+09	6883.07	452.65	6.53766	1.65251	0.125283	0.10092
Functions solved	2	9	14	24	34	36	37

B Benchmark Functions

(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: $A_n(x^*) = 0$

(BE) **Beale Function**
Definition: $BE(x) = (1.5 - x_1 - x_1x_2)^2 + (2.25 - x_1 - x_1x_2^2)^2 + (2,625 - x1 - x_1x_2^3)^2$
Domain: $[-4.5, 4.5]^2$
Global Minimum: $BE(x^*) = 0$

(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: $B2(x^*) = 0$

(BO) **Booth Function**
Definition: $BO(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$
Domain: $[-10, 10]^2$
Global Minimum: $BO(x^*) = 0$

(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: $[-5, 15]$
Global Minimum: $BR(x^*) = 0.397887$

(CV) **Colville Function** (also called **Wood Function**)
Definition: $CV(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 + 90(x_4 - x_3^2)^2 + (1 - x3)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1)$
Domain: $[-10, 10]^4$
Global Minimum: $CV(x^*) = 0$

(DP_n) **Dixon and Price Function**
Definition: $DP_n(x) = (x1 - 1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$
Domain: $[-10, 10]^n$
Global Minimum: $DP_n(x^*) = 0$

(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: $EA(x^*) = -1$

(GP) **Goldstein and Price Function**
Definition: $GP(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)][30 + (2x_1 - 3x_2)^2(18 - 32_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$
Domain: $[-2, 2]^2$
Global Minimum: $GP(x^*) = 3$

(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: $GR_n(x^*) = 0$

($H_{n,m}$) **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: $(n = 3, m = 4)\ H_{3,4}(x^*) = -3.86278$
Global Minimum: $(n = 6, m = 4)\ H_{6,4}(x^*) = -3.32237$
Parameters:

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

$$P^{(3)} = 10^{-4} \begin{bmatrix} 3689 & 1170 & 2673 \\ 4699 & 4387 & 7470 \\ 1091 & 8732 & 5547 \\ 381 & 5743 & 8828 \end{bmatrix}$$

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

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

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

(L_n) **Levy Function**

Definition: $L_n(x) = \sin^2(\pi y_1) + \sum_{i=1}^{n-1}[(y_i - 1)^2(1 + 10\sin^2(\pi y_i + 1))] + (y_n - 1)^2(1 + 10\sin^2(2\pi y_n))$
Domain: $[-10, 10]^n$
Global Minimum: $L_n(x^*) = 0$
Parameters: $y_i = 1 + \frac{x_i - 1}{4}.\ \forall i = 1, \cdots, n$

(M) **Matyas Function**

Definition: $M(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$
Domain: $[-5, 10]^2$
Global Minimum: $M(x^*) = 0$

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

Definition: $P_{n,\beta}(x) = \sum_{k=1}^{n}[\sum_{i=1}^{n}(i^k + \beta)((\frac{x_i}{i})^k - 1)]^2$
Domain: $[-n, n]^n$
Global Minimum: $P_{n,\beta}(x^*) = 0$

($P^0_{n,b}$) **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: $P^0_{n,\beta}(x^*) = 0$

(PW_n) **Powell Function**
Definition: $PW_n(x) = \sum_{i=1}^{\frac{n}{4}}\left[(x_{4i-3}+10x_{4i-2})^2 + 5(x_{4i-1}-x_{4i})^2 + (x_{4i-2}-2x_{4i-1})^4 + 10(x_{4i-3}-x_{4i})^4\right]$
Domain: $[-4,5]^n$
Global Minimum: $PW_n(x^*) = 0$

($PS_{n,b}$) **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 ($n = 4$,$b = \{8,18,44,114\}$): $PS_{4,\{8,18,44,114\}}(x^*) = 0$

(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: $RA_n(x^*) = 0$

(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: $R_n(x^*) = 0$

(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: $SC_n(x^*) = 0$

($S_{4,m}$) **Shekel 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: $S_{4,5}(x^*) = -10.15319538$, $S_{4,7}(x^*) = -10.40281868$, and $S_{4,10}(x^*) = -10.53628349$
Parameters:

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

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

(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: $SH(x^*) = -186.7309$

(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: $CA(x^*) = -1.03162801$

(SP_n) **Sphere Function**
Definition: $SP_n(x) = \sum_{i=1}^{n} x_i^2$
Domain: $[-2.56,5.12]^n$
Global Minimum: $SP_n(x^*) = 0$
N.B.: The De Joung Function (DJ) is 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: $SS_n(x^*) = 0$

(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: $T_6(x^*) = -50$ 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: $Z_n(x^*) = 0$