Université Libre de Bruxelles
Faculté des Sciences Appliquées
CODE - Computers and Decision Engineering
IRIDIA - Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Improved ant colony optimization algorithms for continuous and mixed discrete-continuous optimization problems

Tianjun LIAO

Superviseur:

Prof. Marco DORIGO and Dr. Thomas STÜTZLE

Rapport d'avancement de recherche
Année Académique 2010/2011

IRIDIA
Institut de Recherches Interdisciplinaires
et de Développements en Intelligence Artificielle

Abstract

The interest in using the Ant Colony Optimization (ACO) metaheuristic to solve continuous or mixed discrete-continuous variable optimization problems is increasing. One of the most popular Ant Colony Optimization (ACO) algorithms for the continuous domain is $\mathrm{ACO}_{\mathbb{R}}$. In this thesis, we propose an incremental ant colony algorithm with local search for continuous optimization ($\mathrm{IACO}_{\mathbb{R}}$-LS), and we present an ant colony optimization algorithm for mixed discrete-continuous variable optimization problems ($\mathrm{ACO}_{\mathbf{MV}}$).

We start by a detailed experimental analysis of $\mathrm{ACO}_{\mathbb{R}}$ and based on the obtained insights on $\mathrm{ACO}_{\mathbb{R}}$, we propose $\mathrm{IACO}_{\mathbb{R}}$-LS. This mechanism consists of a growing solution archive with a special initialization rule applied to entrant solutions. The resulting algorithm, called $\mathrm{IACO}_{\mathbb{R}}$, is then hybridized with a local search procedure in order to enhance its search intensification. Automatic parameter tuning results show that $\mathrm{IACO}_{\mathbb{R}}$-LS with Lin-Yu Tseng's Mtsls1 local search algorithm ($\mathrm{IACO}_{\mathbb{R}}$-Mtsls1) significantly outperforms $\mathrm{ACO}_{\mathbb{R}}$, and it is also competitive with state-of-the-art algorithms.

We also show how $\mathrm{ACO}_{\mathbb{R}}$ may be extended to mixed-variable optimization problems. The proposed $\mathrm{ACO}_{\mathbf{MV}}$ algorithm allows to declare each variable of the considered problem as continuous, ordered discrete or categorical discrete. Based on the solution archive framework of $\mathrm{ACO}_{\mathbf{MV}}$, a continuous relaxation approach ($\mathrm{ACO}_{\mathbf{MV}}$-o), a native mixed-variable optimization approach ($\mathrm{ACO}_{\mathbf{MV}}$-c), as well as $\mathrm{ACO}_{\mathbb{R}}$ are integrated to solve continuous and mixed-variable optimization problems. An additional contribution is a new set of artificial mixed-variable benchmark functions, which can simulate discrete variables as ordered or categorical. After automatically tuning the parameters of $\mathrm{ACO}_{\mathbf{MV}}$ on artificial mixed-variable benchmark functions, we test generic $\mathrm{ACO}_{\mathbf{MV}}$ on various real-world continuous and mixed-variable engineering optimization problems. A comparison to results from literature proves $\mathrm{ACO}_{\mathbf{MV}}$'s high performance, and demonstrates its effectiveness and robustness.

Acknowledgements

I thank Prof. Marco Dorigo and Dr. Thomas Stützle for affording me the chance to research at IRIDIA. I also thank Dr. Thomas Stützle and Marco A. Montes de Oca for their guidance and support that help me to conduct the research presented in this work. I also thank Dr. Manuel López-Ibáñez and Jérémie Dubois-Lacoste for their improved implementation of Iterated-F-Race package. Special thanks go to Zhi Yuan and Marco A. Montes de Oca. Talking with them helps me avoid detours, which is very helpful for my work efficiency. I would also like to express my gratitude to Mohamed Saifullah Hussin, Sabrina Oliveira and my colleagues at IRIDIA. Your kindness makes my life so enjoyable here.

This work was supported by E-SWARM project funded ERC Advanced Grant, and supported by Meta-X project funded by the Scientific Research Directorate of the French Community of Belgium.

Statement

This work describes an original research carried out by the author. It has not been previously submitted to any other university for the award of any degree. Nevertheless, some chapters of this thesis are partially based on papers, together with other co-authors, which have been published, submitted or prepared for publication in the scientific literature.

Chapter 3 is based on the paper:

T. Liao, M. Montes de Oca, D. Aydın, T. Stützle, and M. Dorigo. An Incremental Ant Colony Algorithm with Local Search for Continuous Optimization. *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011.* N. Krasnogor et al. (Eds.) Dublin, Ireland. July 2011. Accepted.
Nominated for the best paper award in the Ant Colony Optimization and Swarm Intelligence track (three of four referees nominated it for the best paper award).

Chapter 4 is based on the paper:

T. Liao, K. Socha, M. Montes de Oca, T. Stützle, and M. Dorigo. Ant Colony Optimization for Mixed-Variables Problems. *IEEE Transaction on evolutionary computation.* In preparation for submission.

Other related publications:

T. Liao, M. Montes de Oca, and T. Stützle. Tuning Parameters across Mixed Dimensional Instances: A Performance Scalability Study of Sep-G-CMA-ES. *Proceedings of the Workshop on Scaling Behaviours of Landscapes, Parameters and Algorithms of the Genetic and Evolutionary Computation Conference (GECCO 2011).* E. Özcan, A. J. Parkes, and J. Rowe. (Eds.) Dublin, Ireland. July 2011. Accepted.

T. Liao, M. Montes de Oca, and T. Stützle. A Note on the Effects of Enforcing Bound Constraints on Algorithm Comparisons using the IEEE CEC'05 Benchmark Function Suite, TR/IRIDIA/2011-010, IRIDIA, Université Libre de Bruxelles, Belgium, April 2011.

Contents

Chapter 1

Introduction

Metaheuristics are a family of optimization techniques, which have seen increasingly rapid development and application to numerous problems in computer science and other related fields over the past few years. One of the more recent and actively developed metaheuristics is ant colony optimization (ACO). ACO was inspired by the ants' foraging behavior [24]. It was originally introduced to solve discrete optimization problems [24,25,87], in which each decision variable is characterized by a finite set of components. Many successful implementations of the ACO metaheuristic have been applied to a number of different discrete optimization problems. These applications mainly concern NP-hard combinatorial optimization problems including problems in routing [37], assignment [87], scheduling [86] and bioinformatics [14] problems and many other areas.

Although ACO was proposed for discrete optimization problems, its adaptation to solve continuous optimization problems has taken an increasing attention [10, 32, 67, 83]. This class of optimization problems requires that each decision variable takes a real value from a given domain. Recently, the $\text{ACO}_{\mathbb{R}}$ algorithm has been proposed [83]. It was successfully evaluated on some small dimensional benchmark functions [83] and was applied to the problem of training neural networks for pattern classification in the medical field [82]. However, $\text{ACO}_{\mathbb{R}}$ and other ACO based continuous algorithms were not tested intensively on widely available higher dimensional benchmark such as these of the recent special issue of the Soft Computing journal [46, 64] (Throughout the rest of the report, we will refer to this special issue as SOCO) to compete with other state-of-the-art continuous solvers. The set of algorithms described in SOCO consists of differential evolution algorithms, memetic algorithms, particle swarm optimization algorithms and other types of optimization algorithms [64]. In SOCO, the differential evolution algorithm (DE) [85], the covariance matrix adaptation evolution strategy with increasing population size (G-CMA-ES) [7], and the real-coded CHC algorithm (CHC) [35] are used as the reference algorithms.

It should be noted that no ACO-based algorithms are tested in SOCO.

The aforementioned discrete and continuous optimization problems correspond to discrete variables and continuous variables, respectively. However, many real engineering problems are modeled using a mix of types of decision variables. A common example is a mix of discrete variables and continuous variables. The former usually involve ordering characteristics, categorical characteristics or both of them. Due to the practical relevance of such problems, many mixed-variable optimization algorithms have been proposed, mainly based on Genetic Algorithms [38], Differential Evolution [85], Particle Swarm Optimization [51] and Pattern Search [92]. However, few ACO extensions are used to tackle mixed-variable optimization problems.

In this report, we propose an improved $\mathrm{ACO}_{\mathbb{R}}$ algorithm for the continuous domain, called $\mathrm{IACO}_{\mathbb{R}}$-LS, that is competitive with the state of the art in continuous optimization. We first present $\mathrm{IACO}_{\mathbb{R}}$, which is an $\mathrm{ACO}_{\mathbb{R}}$ with an extra search diversification mechanism that consists of a growing solution archive. Then, we hybridize $\mathrm{IACO}_{\mathbb{R}}$ with a local search procedure in order to enhance its search intensification abilities. We experiment with three local search procedures: Powell's conjugate directions set [76], Powell's BOBYQA [77], and Lin-Yu Tseng's Mtsls1 [93]. An automatic parameter tuning procedure, Iterated F-race [9, 13], is used for the configuration of the investigated algorithms. The best algorithm found after tuning, $\mathrm{IACO}_{\mathbb{R}}$-Mtsls1, obtains results that are as good as the best of the 16 algorithms featured in SOCO. To assess the quality of $\mathrm{IACO}_{\mathbb{R}}$-Mtsls1 and the best SOCO algorithms on problems not seen during their design phase, we compare their performance using an extended benchmark functions suite that includes functions from SOCO and the Special Session on Continuous Optimization of the IEEE 2005 Congress on Evolutionary Computation (CEC 2005). The results show that $\mathrm{IACO}_{\mathbb{R}}$-Mtsls1 can be considered to be a state-of-the-art continuous optimization algorithm.

Next, we present an $\mathrm{ACO}_{\mathbb{R}}$ extension for mixed-variable optimization problems, called $\mathrm{ACO}_{\mathbf{MV}}$. $\mathrm{ACO}_{\mathbf{MV}}$ integrates a component of a continuous relaxation approach ($\mathrm{ACO}_{\mathbf{MV}}$-o) and a component of a native mixed-variable optimization approach ($\mathrm{ACO}_{\mathbf{MV}}$-c), as well as $\mathrm{ACO}_{\mathbb{R}}$ and allows to declare each variable of the mixed variable optimization problems as continuous, ordered discrete or categorical discrete. We also propose a new set of artificial mixed-variable benchmark functions and their constructive methods, thereby providing a flexibly controlled environment for training parameters of mixed-variable optimization algorithms and investigating their performance. We automatically tune the parameters of $\mathrm{ACO}_{\mathbf{MV}}$ on benchmark functions, then compare the performance of $\mathrm{ACO}_{\mathbf{MV}}$ on four classes of eight various mixed-variables engineering optimization problems with the results from literature. $\mathrm{ACO}_{\mathbf{MV}}$ has efficiently found all the best-so-far solution including two new best solution. $\mathrm{ACO}_{\mathbf{MV}}$ obtains a 100% success rate in seven problems. In five of these seven problems, $\mathrm{ACO}_{\mathbf{MV}}$ reaches them in the smallest function evaluations. When compared to 26 other algorithms, $\mathrm{ACO}_{\mathbf{MV}}$ has the best performance on mixed-variables engineering optimization problems from the literature.

The thesis is organized as follows. Chapter 2 introduces the basic principle of the ACO metaheuristic and the $\mathrm{ACO}_{\mathbb{R}}$ algorithm for the continuous domains. In chapter 3, we propose the $\mathrm{IACO}_{\mathbb{R}}$-LS algorithm, which is competitive with state-of-the-art algorithms for continuous optimization. In Chapter 4, we show how $\mathrm{ACO}_{\mathbb{R}}$ may be extended to mixed-variable optimization problems and we propose the $\mathrm{ACO}_{\mathbf{MV}}$ algorithm. We also propose a new set of artificial mixed-variable benchmark functions, which can simulate discrete variables as ordered or categorical. The experimental comparison to results from literature proves $\mathrm{ACO}_{\mathbf{MV}}$'s high performance. In Chapter 5, we summarize some conclusions and directions for future work.

Chapter 2

Ant Colony Optimization

Ant Colony Optimization (ACO) algorithms are constructive stochastic search procedures that make use of a pheromone model and heuristic information on the problem being tackled in order to probabilistically construct solutions. A pheromone model is a set of so-called pheromone trail parameters. The numerical values of these pheromone trail parameters reflect the search experience of the algorithm. They are used to bias the solution construction over time towards the regions of the search space containing high quality solutions. The stochastic procedure in ACO algorithms allows the ants to explore a much larger number of solutions, meanwhile, the use of heuristic information guides the ants towards the most promising solutions. The ants' search experience is to influence the solution construction in future iterations of the algorithm by a reinforcement type of learning mechanism [89].

Ant System (AS) was proposed as the first ACO algorithm for the well known traveling salesman problem (TSP) [30]. Despite AS was not competitive with state-of-the-art algorithms on the TSP, it stimulated further research on algorithmic variants for better computational performance. Several improved ACO algorithms [31] for NP-hard problems that have been proposed in the literature. Ant Colony System (ACS) [29] and $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) algorithm [87] are among the most successful ACO variants in practice. For providing a unifying view to identify the most important aspects of these algorithms, Dorigo et al. [28] put them in a common framework by defining the Ant Colony Optimization (ACO) meta-heuristic. The outline of the ACO metaheuristic [28] is shown in Algorithm 1. After initializing parameters and pheromone trails, the metaheuristic iterates over three phases: at each iteration, a number of solutions are constructed by the ants; these solutions are then improved through a local search (this step is optional), and finally the pheromone trails are updated.

In ACO for combinatorial problems, the pheromone values are associated with a finite set of discrete values related to the decisions that the ants make. This is not possible in the continuous and mixed continuous-discrete

Algorithm 1 Outline of ant colony optimization metaheuristic

```
Set parameters, initialize pheromone trails
while termination criterion not satisfied do
  ConstructAntSolution
  ApplyLocalSearch /*optional*/
  Update pheromones
end while
```

variables cases. Thus, applying the ACO metaheuristic to continuous domains is not straightforward. The simplest approach would be to divide the domain of each variable into a set of intervals. A real value is then rounded to the closest bound of its corresponding interval in a solution construction process. This approach has been successfully followed when applying ACO to the protein ligand docking problem [53]. However, when the domain of the variables is large and the required accuracy is high, this approach is not viable [27]. Except this approach, there have been some other attempts to apply ACO-inspired algorithms to continuous optimization problems [33, 62, 67, 75]. The proposed methods often took inspiration from some type of ant behaviors, but did not follow the ACO metaheuristic closely. For this reason, An ACO-inspired algorithm named $\text{ACO}_{\mathbb{R}}$ [83] was proposed, which can handle continuous variables natively. $\text{ACO}_{\mathbb{R}}$ is an algorithm that conceptually directly follows the ideas underlying the ACO metaheuristic. It is now one of the most popular ACO-based algorithms for continuous domains.

2.1 $\text{ACO}_{\mathbb{R}}$: Ant Colony Optimization for Continuous Domains

The fundamental idea underlying $\text{ACO}_{\mathbb{R}}$ is substituting the discrete probability distributions used in ACO algorithms for combinatorial problems with probability density functions in the solution construction phase. To do so, the $\text{ACO}_{\mathbb{R}}$ algorithm stores a set of k solutions, called solution archive, which represents the algorithm's "pheromone model." The solution archive is used to create a probability distribution of promising solutions over the search space. Initially, the solution archive is filled with randomly generated solutions. The algorithm iteratively refines the solution archive by generating m new solutions and then keeping only the best k solutions of the $k + m$ solutions that are available. The k solutions in the archive are always sorted according to their quality (from best to worst). Solutions are generated on a coordinate-per-coordinate basis using mixtures of weighted Gaussian functions. The core of the solution construction procedure is the estimation of multimodal one-dimensional probability density functions (PDFs). The mechanism to do that in ACO$_\mathbb{R}$ is based on a Gaussian kernel, which is defined as a weighted sum of several Gaussian functions g_j^i, where j is a solution index and i is a coordinate index. The Gaussian kernel for coordinate i is:

$$G^i(x) = \sum_{j=1}^{k} \omega_j g_j^i(x) = \sum_{j=1}^{k} \omega_j \frac{1}{\sigma_j^i \sqrt{2\pi}} e^{-\frac{(x-\mu_j^i)^2}{2{\sigma_j^i}^2}}, \tag{2.1}$$

where $j \in \{1, ..., k\}$, $i \in \{1, ..., D\}$ with D being the problem dimensionality, and ω_j is a weight associated with the ranking of solution j in the archive, $rank(j)$. The weight is calculated using a Gaussian function:

$$\omega_j = \frac{1}{qk\sqrt{2\pi}} e^{\frac{-(rank(j)-1)^2}{2q^2k^2}}, \tag{2.2}$$

where q is a parameter of the algorithm.

During the solution generation process, each coordinate is treated independently. First, an archive solution is chosen with a probability proportional to its weight. Then, the algorithm samples around the selected solution component s_j^i using a Gaussian PDF with $\mu_j^i = s_j^i$, and σ_j^i equal to

$$\sigma_j^i = \xi \sum_{r=1}^{k} \frac{|s_r^i - s_j^i|}{k-1}, \tag{2.3}$$

which is the average distance between the i-th variable of the solution s_j and the i-th variable of the other solutions in the archive, multiplied by a parameter ξ. The solution generation process is repeated m times for each dimension $i = 1, ..., D$. An outline of ACO$_\mathbb{R}$ is given in Algorithm 2. In ACO$_\mathbb{R}$, due to the specific way the pheromone is represented (i.e., as the solution archive), it is in fact possible to take into account the correlation between the decision variables. A non-deterministic adaptive method is presented in [83]. Each ant chooses a *direction* in the search space at each step of the construction process. The direction is chosen by randomly selecting a solution s_d that is reasonably far away from the solution s_j chosen earlier as the mean of the Gaussian PDF. Then, the vector $\vec{s_j s_d}$ becomes the chosen direction. The probability of choosing solution s_u at step i is the following:

$$p(s_d|s_j)_i = \frac{d(s_d, s_j)_i^4}{\sum_{r=1}^{k} d(s_r, s_j)_i^4} \tag{2.4}$$

where the function $d(.,.)_i$ returns the Euclidean distance in the $(n-i+1)$-dimensional search sub-space [1] between two solutions of the archive T. Once this vector is chosen, the new orthogonal basis for the ant's coordinate system is created using the Gram-Schmidt process [39]. Then, all the current

[1] At step i, only dimensions i through n are used.

Algorithm 2 Outline of $\mathrm{ACO}_{\mathbb{R}}$

Input: k, m, D, q, ξ, and termination criterion.
Output: The best solution found

```
Initialize and evaluate k solutions
// Sort solutions and store them in the archive
T = Sort(S_1 ··· S_k)
while Termination criterion is not satisfied do
  // Generate m new solutions
  for l = 1 to m do
    // Construct solution
    for i = 1 to D do
      Select Gaussian g_j^i according to weights
      Sample Gaussian g_j^i with parameters μ_j^i, σ_j^i
    end for
    Store and evaluate newly generated solution
  end for
  // Sort solutions and select the best k
  T = Best(Sort(S_1 ··· S_{k+m}), k)
end while
```

coordinates of all the solutions in the archive are rotated and recalculated according to this new orthogonal base. At the end of the solution construction process, the chosen values of the temporary variables are converted back into the original coordinate system.

2.2 Further Investigation on $\mathrm{ACO}_{\mathbb{R}}$

The original implementation of $\mathrm{ACO}_{\mathbb{R}}$ is in R [48] that is a language and environment for statistical computing and graphics. For a higher execution efficiency, we developed a C++ implementation. Figure 2.1 and 2.2 shows the coincident performance of two different implementation and illustrates the validity of C++ implementation. Moreover, Figure 2.1 is shown on both non-rotated and rotated functions to demonstrate the positive effect of variable correlation handling method in $\mathrm{ACO}_{\mathbb{R}}$ on rotated functions. The formula of the non-rotated and rotated functions are given in Table 2.1.

When $\mathrm{ACO}_{\mathbb{R}}$ constructs a solution, a Gram-Schmidt process is used for the new orthogonal basis of the ant's coordinate system. Although, it helps to handle variable correlation, the calculation of the Gram-Schmidt process for each variable for each constructive step incurs a very high computational demand. When the dimension of the objective function increases, the time used by $\mathrm{ACO}_{\mathbb{R}}$ with this variable correlation handling increases rapidly. For this reason, we also developed a C++ implementation of $\mathrm{ACO}_{\mathbb{R}}$ that does not consider variable correlation handling part (it is referred as Sep-$\mathrm{ACO}_{\mathbb{R}}$

Table 2.1: The formula of the non-rotated and rotated benchmark functions for Figure 2.1

The objective functions
$ellipsoid(\vec{x}) = \sum_{i=1}^{n}(100^{\frac{i-1}{n-1}} x_i)^2$
$rotatedellipsoid(\vec{x}) = \sum_{i=1}^{n}(100^{\frac{i-1}{n-1}} z_i)^2$
$tablet(\vec{x}) = 10^4 {x_1}^2 + \sum_{i=2}^{n} {x_i}^2$
$rotatedtablet(\vec{x}) = 10^4 {z_1}^2 + \sum_{i=2}^{n} {z_i}^2$
$cigar(\vec{x}) = {x_1}^2 + 10^4 \sum_{i=2}^{n} {x_i}^2$
$rotatedcigar(\vec{x}) = {z_1}^2 + 10^4 \sum_{i=2}^{n} {z_i}^2$
The definition of the variables
$\vec{z} = (\vec{x})\mathbf{M}, \vec{x} \in (-3, 7)$
$\mathbf{M}$ is a random, normalized n-dimensional rotation matrix

throughout the rest of this thesis).

To illustrate the execution time issue of $\text{ACO}_{\mathbb{R}}$, Figure 2.3 shows the average execution time of Sep-$\text{ACO}_{\mathbb{R}}$ and $\text{ACO}_{\mathbb{R}}$ in dependence of the number of dimensions of the problem after 1000 evaluations. We fitted quadratic functions to the observed computation times. As seen from Figure 2.3, the fitted model for Sep-$\text{ACO}_{\mathbb{R}}$ can be treated as linear due to the tiny coefficient for the quadratic term. The execution time of $\text{ACO}_{\mathbb{R}}$ scales quadratically with the dimensions of the testing problems. Taking the Rastrigin benchmark function for example, the execution time of $\text{ACO}_{\mathbb{R}}$ that corresponds to 40 dimensions after 1000 function evaluations is about 50 seconds. Since the time cost of each function evaluation is similar, $\text{ACO}_{\mathbb{R}}$ need about 9 hours for 1000 dimensional Rastrigin function every 1000 evaluations. We can predict that $\text{ACO}_{\mathbb{R}}$ need about 5 years for 1000 dimensional rastrigin function after 5000*D (D=1000) evaluations, which is the termination criterion for the large scale continuous optimization benchmark problems [46]. Sep-$\text{ACO}_{\mathbb{R}}$ only needs about 5 seconds and 7 hours for 1000 dimensional rastrigin function after 1000 evaluations and 5000*D (D=1000) evaluations, respectively. With this variable correlation handling method, $\text{ACO}_{\mathbb{R}}$ is difficult and infeasible to apply to higher dimensional optimization problems. Therefore, Sep-$\text{ACO}_{\mathbb{R}}$ is usually substituted for $\text{ACO}_{\mathbb{R}}$ and it is extended to apply to many large scale optimization problems.

Figures 2.4 and 2.5 show the performance of Sep-$\text{ACO}_{\mathbb{R}}$ with different parameter configurations on selected 100 dimensional benchmark functions. We investigate the four different combinations of parameters q and k in the $\text{ACO}_{\mathbb{R}}$ algorithm ($q \in (0.0001, 0.1)$, $k \in (50, 100)$), with the default parameters $m = 2$ and $\xi = 0.85$. Although Sep-$\text{ACO}_{\mathbb{R}}$ has shown a good performance for continuous domains with certain parameter configurations, the gap with the-state-of-art continuous solvers is still considerable, which is shown in the following Chapter 3. The different performances caused by different parameter configurations in Figures 2.4 and 2.5 indicate that a automatic parameter tuning method may help to improve the performance of Sep-ACO$_{\mathbb{R}}$. In the following Chapter 3, an improved ACO algorithm is presented.

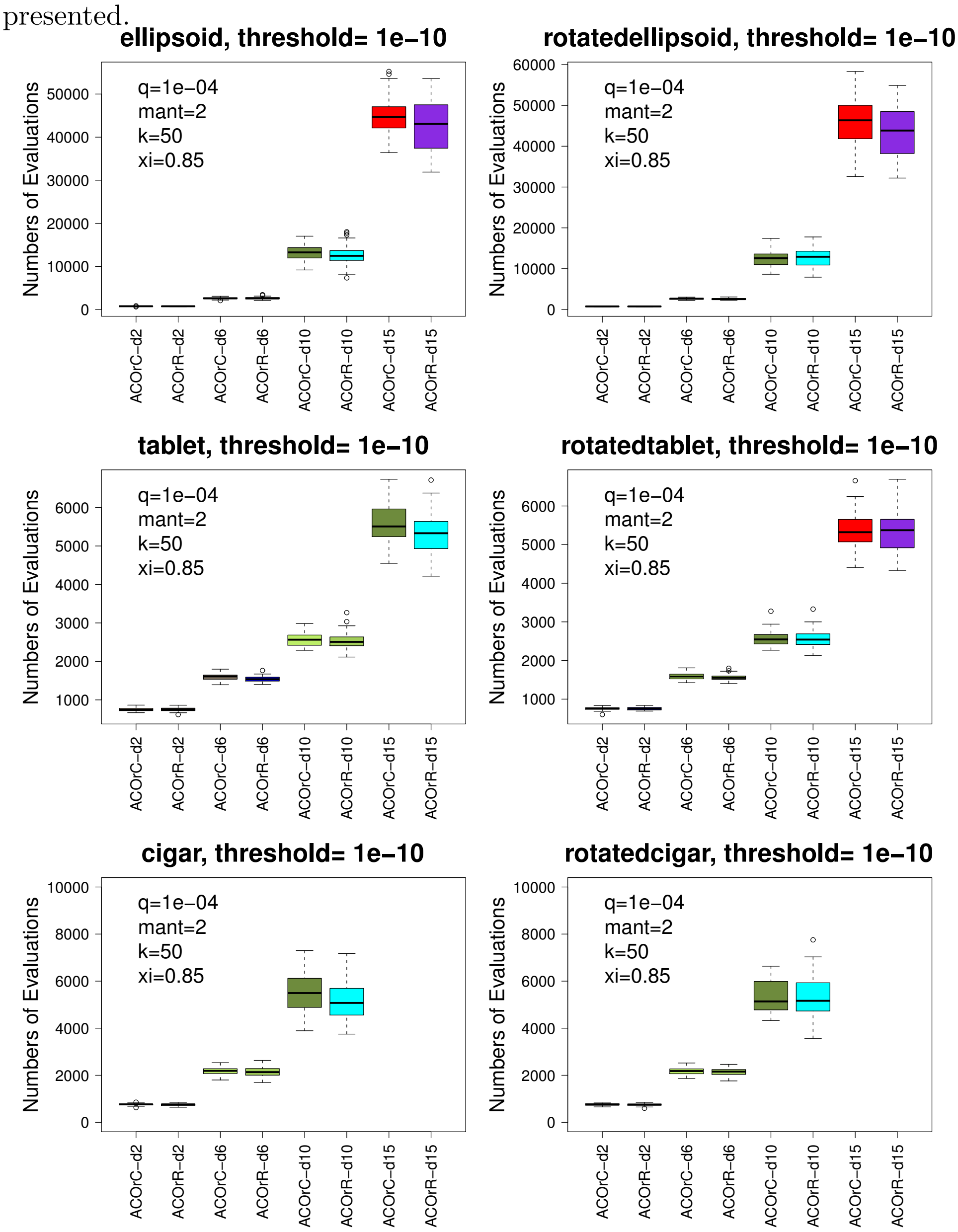

Figure 2.1: The box-plot comparison between C++ and R implementation of ACO$_{\mathbb{R}}$ on different dimensionality. ACOrC is the C++ implementation of ACO$_{\mathbb{R}}$ and ACOrR is the original R implementation. The left box plots are shown the numbers of function evaluations when achieving the threshold of solution quality in non-rotated functions, on the right box-plots those of rotated functions. We set the threshold of solution quality to 1e−10. The adopted parameter configurations of ACO$_{\mathbb{R}}$ are shown in the legends.

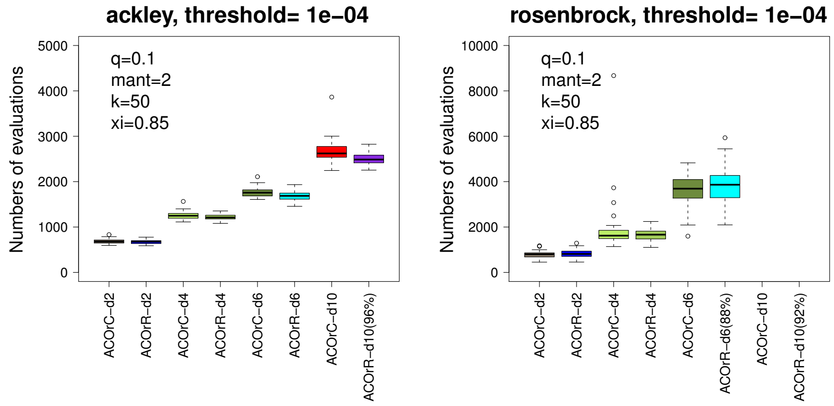

Figure 2.2: The box-plot comparison of C++ and R implementation of ACO$_\mathbb{R}$ on different dimensionality of benchmark functions. ACOrC is the C++ implementation of ACO$_\mathbb{R}$ and ACOrR is the original R implementation. The box plots are shown the numbers of function evaluations when achieving the threshold of solution quality. We set the threshold of solution quality to 1e−04 for ackley and rosenbrock functions.

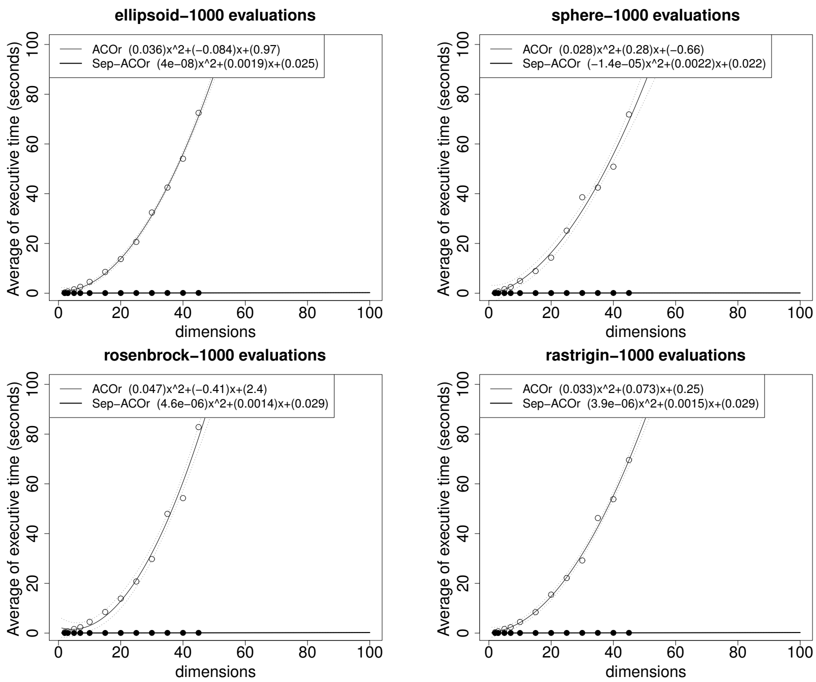

Figure 2.3: The fitted functions show the average time of Sep-ACO$_\mathbb{R}$ and ACO$_\mathbb{R}$ after 1000 functions evaluations in dependence of the increasing dimensionality. The fitted functions are modeled using the execution time on 2, 3, 5, 7, 10, 15, 20, 25, 30, 35, 40 and 45 dimensions. The function expressions are shown in the legends.

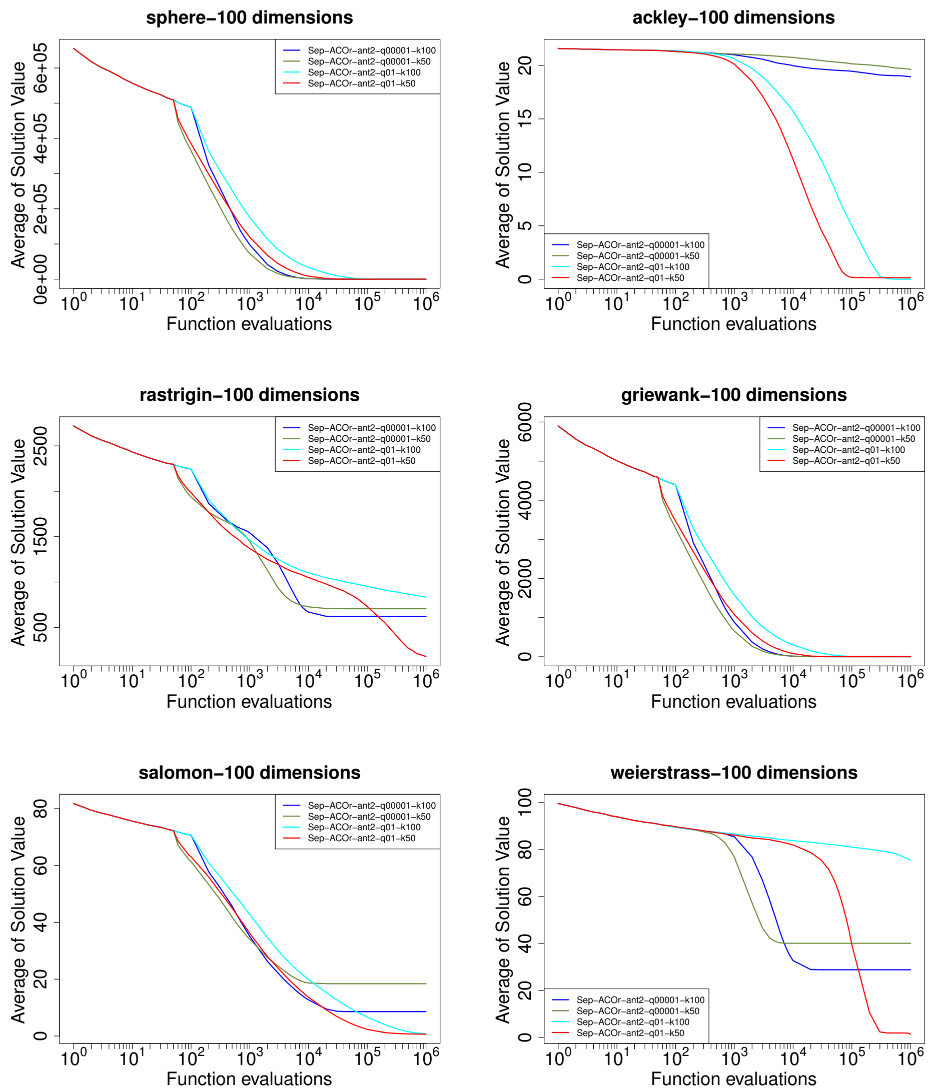

Figure 2.4: The development of the solution quality over the number of function evaluations for Sep-ACO$_{\mathbb{R}}$ with different parameter configurations. The adopted parameter configurations for Sep-ACO$_{\mathbb{R}}$ are shown in the legends.

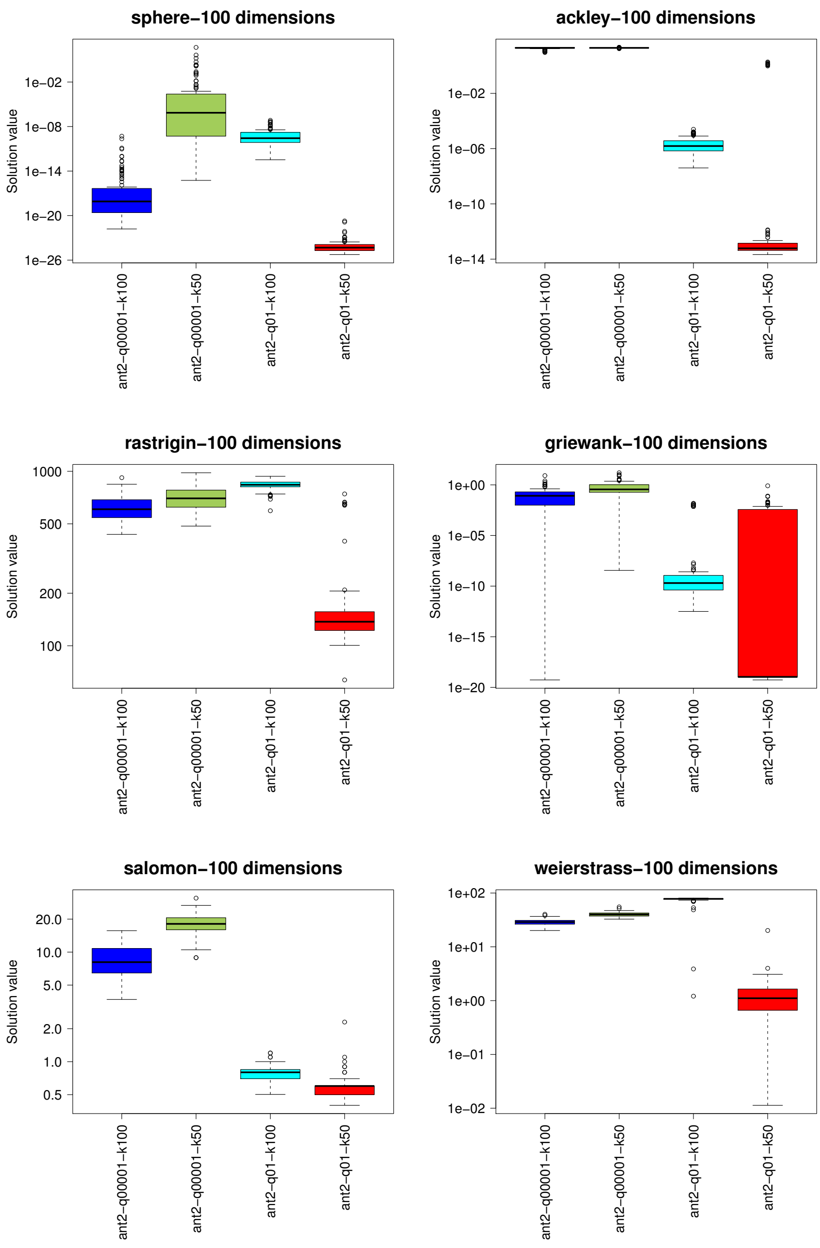

Figure 2.5: The box-plots show the solution quality after 1E+06 function evaluations for Sep-ACO$_{\mathbb{R}}$ with different parameter configurations.

Chapter 3

An Incremental Ant Colony Algorithm with Local Search

As we have seen in Chapter 2, $ACO_{\mathbb{R}}$ [83] and Sep-$ACO_{\mathbb{R}}$ were further evaluated and analyzed on selected benchmark functions. The main drawbacks are the high execution time cost for $ACO_{\mathbb{R}}$ and the performance gap with the-state-of-art continuous solvers, respectively. How to improve $ACO_{\mathbb{R}}$ is therefore an important work. Recently, Leguizamón and Coello [57] proposed a variant of $ACO_{\mathbb{R}}$ that performs better than the original $ACO_{\mathbb{R}}$ on six benchmark functions. However, the results obtained with Leguizamón and Coello's variant are far from being competitive with the results obtained by state-of-the-art continuous optimization algorithms recently featured in a special issue of the Soft Computing journal [64] (Throughout the rest of this chapter, we will refer to this special issue as SOCO). The set of algorithms described in SOCO consists of differential evolution algorithms, memetic algorithms, particle swarm optimization algorithms and other types of optimization algorithms [64]. In SOCO, the differential evolution algorithm (DE) [85], the covariance matrix adaptation evolution strategy with increasing population size (G-CMA-ES) [7], and the real-coded CHC algorithm (CHC) [35] are used as the reference algorithms. It should be noted that no ACO-based algorithms are featured in SOCO.

In this chapter, we propose an improved $ACO_{\mathbb{R}}$ algorithm, called I$ACO_{\mathbb{R}}$-LS, that is competitive with the state of the art in continuous optimization. We first present I$ACO_{\mathbb{R}}$, which is an $ACO_{\mathbb{R}}$ with an extra search diversification mechanism that consists of a growing solution archive. Then, we hybridize I$ACO_{\mathbb{R}}$ with a local search procedure in order to enhance its search intensification abilities. We experiment with three local search procedures: Powell's conjugate directions set [76], Powell's BOBYQA [77], and Lin-Yu Tseng's Mtsls1 [93]. An automatic parameter tuning procedure, Iterated F-race [9, 13], is used for the configuration of the investigated algorithms. The best algorithm found after tuning, I$ACO_{\mathbb{R}}$-Mtsls1, obtains results that are as good as the best of the 16 algorithms featured in SOCO. To assess the quality of IACO$_\mathbb{R}$-Mtsls1 and the best SOCO algorithms on problems not seen during their design phase, we compare their performance using an extended benchmark functions suite that includes functions from SOCO and the special session on continuous optimization of the IEEE 2005 Congress on Evolutionary Computation (CEC 2005). The results show that IACO$_\mathbb{R}$-Mtsls1 can be considered to be a state-of-the-art continuous optimization algorithm.

3.1 The IACO$_\mathbb{R}$ Algorithm

IACO$_\mathbb{R}$ is an ACO$_\mathbb{R}$ algorithm with a solution archive whose size increases over time. This modification is based on the incremental social learning framework [70, 72]. A parameter *Growth* controls the rate at which the archive grows. Fast growth rates encourage search diversification while slow ones encourage intensification [70]. In IACO$_\mathbb{R}$ the optimization process begins with a small archive, a parameter *InitArchiveSize* defines its size. A new solution is added to it every *Growth* iterations until a maximum archive size, denoted by *MaxArchiveSize*, is reached. Each time a new solution is added, it is initialized using information from the best solution in the archive. First, a new solution $\boldsymbol{S}_{\text{new}}$ is generated completely at random. Then, it is moved toward the best solution in the archive $\boldsymbol{S}_{\text{best}}$ using

$$\boldsymbol{S}'_{\text{new}} = \boldsymbol{S}_{\text{new}} + \text{rand}(0,1)(\boldsymbol{S}_{\text{best}} - \boldsymbol{S}_{\text{new}})\,, \tag{3.1}$$

where $\text{rand}(0,1)$ is a random number in the range $[0,1)$.

IACO$_\mathbb{R}$ also features a mechanism different from the one used in the original ACO$_\mathbb{R}$ for selecting the solution that guides the generation of new solutions. The new procedure depends on a parameter $p \in [0,1]$, which controls the probability of using only the best solution in the archive as a guiding solution. With a probability $1-p$, all the solutions in the archive are used to generate new solutions. Once a guiding solution is selected, and a new one is generated (in exactly the same way as in ACO$_\mathbb{R}$), they are compared. If the newly generated solution is better than the guiding solution, it replaces it in the archive. This replacement strategy is different from the one used in ACO$_\mathbb{R}$ in which all the solutions in the archive and all the newly generated ones compete.

We include an algorithm-level diversification mechanism for fighting stagnation. The mechanism consists in restarting the algorithm and initializing the new initial archive with the best-so-far solution. The restart criterion is the number of consecutive iterations, $MaxStagIter$, with a relative solution improvement lower than a certain threshold.

3.2 IACO$_\mathbb{R}$ with Local Search

The IACO$_\mathbb{R}$-LS algorithm is a hybridization of IACO$_\mathbb{R}$ with a local search procedure. IACO$_\mathbb{R}$ provides the exploration needed to locate promising solutions and the local search procedure enables a fast convergence toward good solutions. In our experiments, we considered Powell's conjugate directions set [76], Powell's BOBYQA [77] and Lin-Yu Tseng's Mtsls1 [93] methods as local search procedures. We used the NLopt library [49] implementation of the first two methods and implemented Mtsls1 following the pseudocode found in [93].

In IACO$_\mathbb{R}$-LS, the local search procedure is called using the best solution in the archive as initial point. The local search methods terminate after a maximum number of iterations, *MaxITER*, have been reached, or when the tolerance, that is the relative change between solutions found in two consecutive iterations, is lower than a parameter *FTOL*. Like [71], we use an adaptive step size for the local search procedures. This is achieved as follows: a solution in the archive, different from the best solution, is chosen at random. The maximum norm ($||\cdot||_\infty$) of the vector that separates this random solution from the best solution is used as the local search step size. Hence, step sizes tend to decrease over time due to the convergence tendency of the solutions in the archive. This phenomenon in turn makes the search focus around the best-so-far solution.

For fighting stagnation at the level of the local search, we call the local search procedure from different solutions from time to time. A parameter, *MaxFailures*, determines the maximum number of repeated calls to the local search method from the same initial solution that does not result in a solution improvement. We maintain a failures counter for each solution in the archive. When a solution's failures counter is greater than or equal to *MaxFailures*, the local search procedure is not called again from this solution. Instead, the local search procedure is called from a random solution whose failures counter is less than *MaxFailures*.

Finally, we use a simple mechanism to enforce boundary constraints in IACO$_\mathbb{R}$-LS. We use the following penalty function in Powell's conjugate directions method as well as in Mtsls1:

$$P(\boldsymbol{x}) = fes \cdot \sum_{i=1}^{D} Bound(x_i)\,, \tag{3.2}$$

where $Bound(x_i)$ is defined as

$$Bound(x_i) = \begin{cases} 0, & \text{if } x_{\min} \leq x_i \leq x_{\max} \\ (x_{\min} - x_i)^2, & \text{if } x_i < x_{\min} \\ (x_{\max} - x_i)^2, & \text{if } x_i > x_{\max} \end{cases} \tag{3.3}$$

where x_{min} and x_{max} are the minimum and maximum limits of the search range, respectively, and *fes* is the number of function evaluations that have been used so far. BOBYQA has its own mechanism for dealing with bound constraints. IACO$_{\mathbb{R}}$-LS is shown in Algorithm 3. The C++ implementation of IACO$_{\mathbb{R}}$-LS is available in `http://iridia.ulb.ac.be/supp/IridiaSupp2011-008/`.

3.3 Experimental Study

Our study is carried out in two stages. First, we evaluate the performance of ACO$_{\mathbb{R}}$, IACO$_{\mathbb{R}}$-BOBYQA, IACO$_{\mathbb{R}}$-Powell and IACO$_{\mathbb{R}}$-Mtsls1 by comparing their performance with that of the 16 algorithms featured in SOCO. For this purpose, we use the same 19 benchmark functions suite (functions labeled as $f_{soco}*$). Second, we include 21[1] of the benchmark functions proposed for the special session on continuous optimization organized for the IEEE 2005 Congress on Evolutionary Computation (CEC 2005) [88] (functions labeled as $f_{cec}*$).

In the first stage of the study, we used the 50- and 100-dimensional versions of the 19 SOCO functions. Functions f_{soco1}–f_{soco6} were originally proposed for the special session on large scale global optimization organized for the IEEE 2008 Congress on Evolutionary Computation (CEC 2008) [90]. Functions f_{soco7}-f_{soco11} were proposed at the ISDA 2009 Conference. Functions f_{soco12}-f_{soco19} are hybrid functions that combine two functions belonging to f_{soco1}–f_{soco11}. The detailed description of these functions is available in [46, 64]. In the second stage of our study, the 19 SOCO and 21 CEC 2005 functions on 50 dimensions were considered together. Some properties of the benchmark functions are listed in Table 3.1. The detailed description is available in [46, 88].

We applied the termination conditions used for SOCO and CEC 2005 were used, that is, the maximum number of function evaluations was $5000 \times D$ for the SOCO functions, and $10000 \times D$ for the CEC 2005 functions. All the investigated algorithms were run 25 times on each function. We report error values defined as $f(\boldsymbol{x}) - f(\boldsymbol{x}^*)$, where $\boldsymbol{x}$ is a candidate solution and $\boldsymbol{x}^*$ is the optimal solution. Error values lower than 10^{-14} (this value is referred to as *0-threshold*) are approximated to 0. Our analysis is based on either the whole solution quality distribution, or on the median and average errors.

3.3.1 Parameter Settings

We used Iterated F-race [9, 13] to automatically tune algorithm parameters. The 10-dimensional versions of the 19 SOCO functions were randomly sampled as training instances. A maximum of 50,000 algorithm runs were used as tuning budget for ACO$_{\mathbb{R}}$, IACO$_{\mathbb{R}}$-BOBYQA, IACO$_{\mathbb{R}}$-Powell and

[1] From the original 25 functions, we decided to omit f_{cec1}, f_{cec2}, f_{cec6}, and f_{cec9} because they are the same as f_{soco1}, f_{soco3}, f_{soco4}, f_{soco8}.

Algorithm 3 Outline of IACO$_{\mathbb{R}}$-LS

Input: : ξ, p, *InitArchiveSize*, *Growth*, *MaxArchiveSize*, *FTOL*, *MaxITER*, *MaxFailures*, *MaxStagIter*, D and termination criterion.
Output: The best solution found
  $k = InitArchiveSize$
  Initialize and evaluate k solutions
  while Termination criterion not satisfied **do**
    // Local search
    if $FailedAttempts_{\text{best}} < MaxFailures$ **then**
      Invoke local search from $\boldsymbol{S}_{\text{best}}$ with parameters *FTOL* and *MaxITER*
    else
      if $FailedAttempts_{\text{random}} < MaxFailures$ **then**
        Invoke local search from $\boldsymbol{S}_{\text{random}}$ with parameters *FTOL* and *MaxITER*
      end if
    end if
    if No solution improvement **then**
      $FailedAttempts_{\text{best}||\text{random}} + +$
    end if

    // Generate new solutions
    if rand(0,1)<p **then**
      for $i = 1$ to D **do**
        Select Gaussian g^i_{best}
        Sample Gaussian g^i_{best} with parameters $\mu^i_{\text{best}}, \sigma^i_{\text{best}}$
      end for
      if Newly generated solution is better than $\boldsymbol{S}_{\text{best}}$ **then**
        Substitute newly generated solution for $\boldsymbol{S}_{\text{best}}$
      end if
    else
      for $j = 1$ to k **do**
        for $i = 1$ to D **do**
          Select Gaussian g^i_j
          Sample Gaussian g^i_j with parameters μ^i_j, σ^i_j
        end for
        if Newly generated solution is better than $\boldsymbol{S}_j$ **then**
          Substitute newly generated solution for $\boldsymbol{S}_j$
        end if
      end for
    end if

    // Archive Growth
    if current iterations are multiple of *Growth* & $k < MaxArchiveSize$ **then**
      Initialize new solution using Eq.3.1
      Add new solution to the archive
      $k + +$
    end if
    // Restart Mechanism
    if # of iterations without improving $\boldsymbol{S}_{\text{best}} = MaxStagIter$ **then**
      Re-initialize T but keeping $\boldsymbol{S}_{\text{best}}$
    end if
  end while

Table 3.1: Benchmark functions

ID	Name/Description	Range $[X_{\min}, X_{\max}]^D$	Uni/Multi modal	Separable	Rotated
f_{soco1}	Shift.Sphere	$[-100,100]^D$	U	Y	N
f_{soco2}	Shift.Schwefel 2.21	$[-100,100]^D$	U	N	N
f_{soco3}	Shift.Rosenbrock	$[-100,100]^D$	M	N	N
f_{soco4}	Shift.Rastrigin	$[-5,5]^D$	M	Y	N
f_{soco5}	Shift.Griewank	$[-600,600]^D$	M	N	N
f_{soco6}	Shift.Ackley	$[-32,32]^D$	M	Y	N
f_{soco7}	Shift.Schwefel 2.22	$[-10,10]^D$	U	Y	N
f_{soco8}	Shift.Schwefel 1.2	$[-65.536,65.536]^D$	U	N	N
f_{soco9}	Shift.Extended f_{10}	$[-100,100]^D$	U	N	N
f_{soco10}	Shift.Bohachevsky	$[-15,15]^D$	U	N	N
f_{soco11}	Shift.Schaffer	$[-100,100]^D$	U	N	N
f_{soco12}	$f_{soco9} \oplus_{0.25} f_{soco1}$	$[-100,100]^D$	M	N	N
f_{soco13}	$f_{soco9} \oplus_{0.25} f_{soco3}$	$[-100,100]^D$	M	N	N
f_{soco14}	$f_{soco9} \oplus_{0.25} f_{soco4}$	$[-5,5]^D$	M	N	N
f_{soco15}	$f_{soco10} \oplus_{0.25} f_{soco7}$	$[-10,10]^D$	M	N	N
f_{soco16}	$f_{soco9} \oplus_{0.5} f_{soco1}$	$[-100,100]^D$	M	N	N
f_{soco17}	$f_{soco9} \oplus_{0.75} f_{soco3}$	$[-100,100]^D$	M	N	N
f_{soco18}	$f_{soco9} \oplus_{0.75} f_{soco4}$	$[-5,5]^D$	M	N	N
f_{soco19}	$f_{soco10} \oplus_{0.75} f_{soco7}$	$[-10,10]^D$	M	N	N
f_{cec3}	Shift.Ro.Elliptic	$[-100,100]^D$	U	N	Y
f_{cec4}	Shift.Schwefel 1.2 Noise	$[-100,100]^D$	U	N	N
f_{cec5}	Schwefel 2.6 Opt on Bound	$[-100,100]^D$	U	N	N
f_{cec7}	Shift.Ro.Griewank No Bound	$[0,600]^{D\dagger}$	M	N	Y
f_{cec8}	Shift.Ro.Ackley Opt on Bound	$[-32,32]^D$	M	N	Y
f_{cec10}	Shift.Ro.Rastrigin	$[-5,5]^D$	M	N	Y
f_{cec11}	Shift.Ro.Weierstrass	$[-0.5,0.5]^D$	M	N	Y
f_{cec12}	Schwefel 2.13	$[-\pi,\pi]^D$	M	N	N
f_{cec13}	Griewank plus Rosenbrock	$[-3,1]^D$	M	N	N
f_{cec14}	Shift.Ro.Exp.Scaffer	$[-100,100]^D$	M	N	Y
f_{cec15}	Hybrid Composition	$[-5,5]^D$	M	N	N
f_{cec16}	Ro. Hybrid Composition	$[-5,5]^D$	M	N	Y
f_{cec17}	Ro. Hybrid Composition	$[-5,5]^D$	M	N	Y
f_{cec18}	Ro. Hybrid Composition	$[-5,5]^D$	M	N	Y
f_{cec19}	Ro. Hybrid Composition	$[-5,5]^D$	M	N	Y
f_{cec20}	Ro. Hybrid Composition	$[-5,5]^D$	M	N	Y
f_{cec21}	Ro. Hybrid Composition	$[-5,5]^D$	M	N	Y
f_{cec22}	Ro. Hybrid Composition	$[-5,5]^D$	M	N	Y
f_{cec23}	Ro. Hybrid Composition	$[-5,5]^D$	M	N	Y
f_{cec24}	Ro. Hybrid Composition	$[-5,5]^D$	M	N	Y
f_{cec25}	Ro. Hybrid Composition	$[2,5]^{D\dagger}$	M	N	Y

† denotes initialization range instead of bound constraints.

IACO$_{\mathbb{R}}$-Mtsls1. The number of function evaluations used in each run is equal to 50,000. The best set of parameters, for each algorithm found with this process is given in Table 3.2. The only parameter that we set manually was *MaxArchiveSize*, which we set to 1,000.

Table 3.2: Best parameter settings found through iterated F-Race for $ACO_{\mathbb{R}}$, $IACO_{\mathbb{R}}$-BOBYQA, $IACO_{\mathbb{R}}$-Powell and $IACO_{\mathbb{R}}$-Mtsls1. The parameter $FTOL$ is first transformed as 10^{FTOL} before using it in the algorithms.

$ACO_{\mathbb{R}}$	q	ξ	m	k				
	0.04544	0.8259	10	85				
$IACO_{\mathbb{R}}$-BOBYQA	p	ξ	*InitArchiveSize*	*Growth*	*FTOL*	*MaxITER*	*MaxFailures*	*MaxStagIter*
	0.6979	0.8643	4	1	-3.13	240	5	20
$IACO_{\mathbb{R}}$-Powell	p	ξ	*InitArchiveSize*	*Growth*	*FTOL*	*MaxITER*	*MaxFailures*	*MaxStagIter*
	0.3586	0.9040	1	7	-1	20	6	8
$IACO_{\mathbb{R}}$-Mtsls1	p	ξ	*InitArchiveSize*	*Growth*	*MaxITER*	*MaxFailures*	*MaxStagIter*	
	0.6475	0.7310	14	1	85	4	13	

3.3.2 Experimental Results and Comparison

Figure 3.1 shows the distribution of median and average errors across the 19 SOCO benchmark functions obtained with $ACO_{\mathbb{R}}$, $IACO_{\mathbb{R}}$-BOBYQA, $IACO_{\mathbb{R}}$-Powell, $IACO_{\mathbb{R}}$-Mtsls1 and the 16 algorithms featured in SOCO.[2] We marked with a + symbol those cases in which there is a statistically significant difference at the 0.05 α-level with a Wilcoxon test with respect to $IACO_{\mathbb{R}}$-Mtsls1 (in favor of $IACO_{\mathbb{R}}$-Mtsls1). Also at the top of each plot, a count of the number of optima found by each algorithm (or an objective function value lower than 10^{-14}) is given.

In all cases, $IACO_{\mathbb{R}}$-Mtsls1 significantly outperforms $ACO_{\mathbb{R}}$, and is in general more effective than $IACO_{\mathbb{R}}$-BOBYQA, and $IACO_{\mathbb{R}}$-Powell. $IACO_{\mathbb{R}}$-Mtsls1 is also competitive with the best algorithms in SOCO. If we consider medians only, $IACO_{\mathbb{R}}$-Mtsls1 significantly outperforms G-CMA-ES, CHC, DE, EVoPROpt, VXQR1, EM323, and RPSO-vm in both 50 and 100 dimensions. In 100 dimensions, $IACO_{\mathbb{R}}$-Mtsls1 also significantly outperforms MA-SSW and GODE. Moreover, the median error of $IACO_{\mathbb{R}}$-Mtsls1 is below the 0-threshold 14 times out of the 19 possible of the SOCO benchmark functions suite. Only MOS-DE matches such a performance.

If one considers mean values, the performance of $IACO_{\mathbb{R}}$-Mtsls1 degrades slightly. This is an indication that $IACO_{\mathbb{R}}$-Mtsls1 still stagnates with some low probability. However, $IACO_{\mathbb{R}}$-Mtsls1 still outperforms G-CMA-ES, CHC, GODE, EVoPROpt, RPSO-vm, and EM323. Even though $IACO_{\mathbb{R}}$-Mtsls1 does not significantly outperform DE and other algorithms, its performance is very competitive. The mean error of $IACO_{\mathbb{R}}$-Mtsls1 is below the 0-threshold 13 and 11 times in problems of 50 and 100 dimensions, respectively.

We note that although G-CMA-ES has difficulties in dealing with multimodal or unimodal shifted separable functions, such as f_{soco4} , f_{soco6} and f_{soco7}, G-CMA-ES showed impressive results on function f_{soco8}, which is a hyperellipsoid rotated in all directions. None of the other investigated al-

[2] For information about these 16 algorithms please go to `http://sci2s.ugr.es/eamhco/CFP.php`

gorithms can find the optimum of this function except G-CMA-ES. This result is interesting considering that G-CMA-ES showed an impressive performance in the CEC 2005 special session on continuous optimization. This fact suggests that releasing details about the problems that will be used to compare algorithms induces an undesired “overfitting” effect. In other words, authors may use the released problems to design algorithms that perform well on them but that may perform poorly on another unknown set of problems. This motivated us to carry out the second stage of our study, which consists in carrying out a more comprehensive comparison that includes G-CMA-ES and some of the best algorithms in SOCO. For this comparison, we use 40 benchmark functions as discussed above. From SOCO, we include in our study IPSO-Powell given its good performance as shown in Figure 3.1. To discard the possibility that the local search procedure is the main responsible for the obtained results, we also use Mtsls1 with IPSO, thus generating IPSO-Mtsls1. In this second stage, IPSO-Powell and IPSO-Mtsls1 were tuned as described in Section 3.3.1.

Table 3.4 shows the median and average errors obtained by the compared algorithm on each of the 40 benchmark functions. Two facts can be noticed from these results. First, Mtsls1 seems to be indeed responsible for most of the good performance of the algorithms that use it as a local search procedure. Regarding median results, the SOCO functions for which IPSO-Mtsls1 finds the optimum, IACO$_{\mathbb{R}}$-Mtsls1 does it as well. However, IACO$_{\mathbb{R}}$-Mtsls1 seems to be more robust given the fact that it finds more optima than IPSO-Mtsls1 if functions from the CEC 2005 special session or mean values are considered. Second, G-CMA-ES finds more best results on the CEC 2005 functions than on the SOCO functions. Overall, however, IACO$_{\mathbb{R}}$-Mtsls1 finds more best results than any of the compared algorithms.

Figure 3.2 shows correlation plots that illustrate the relative performance between IACO$_{\mathbb{R}}$-Mtsls1 and G-CMA-ES, IPSO-Powell and IPSO-Mtsls1. On the x-axis, the coordinates are the results obtained with IACO$_{\mathbb{R}}$-Mtsls1; on the y-axis, the coordinates are the results obtained with the other algorithms for each of the 40 functions. Thus, points that appear on the left part of the correlation plot correspond to functions for which IACO$_{\mathbb{R}}$-Mtsls1 has better results than the other algorithm.

Table 3.3 shows a detailed comparison presented in form of (win, draw, lose) according to different properties of the 40 functions used. The two-sided p-values of Wilcoxon matched-pairs signed-ranks test of IACO$_{\mathbb{R}}$-Mtsls1 with other algorithms across 40 functions are also presented. In general, IACO$_{\mathbb{R}}$-Mtsls1 performs better more often than all the other compared algorithms. IACO$_{\mathbb{R}}$-Mtsls1 wins more often against G-CMA-ES; however, G-CMA-ES performs clearly better than IACO$_{\mathbb{R}}$-Mtsls1 on rotated functions, which can be explained by the covariance matrix adaptation mechanism [42].

3.4 Conclusions

In this chapter, we have introduced IACO$_{\mathbb{R}}$-LS, an ACO$_{\mathbb{R}}$ algorithm with growing solution archive hybridized with a local search procedure. Three different local search procedures, Powell's conjugate directions set, Powell's BOBYQA, and Mtsls1, were tested with IACO$_{\mathbb{R}}$-LS. Through automatic tuning across 19 functions, IACO$_{\mathbb{R}}$-Mtsls1 proved to be superior to the other two variants.

The results of a comprehensive experimental comparison with 16 algorithms featured in a recent special issue of the Soft Computing journal show that IACO$_{\mathbb{R}}$-Mtsls1 significantly outperforms the original ACO$_{\mathbb{R}}$ and that IACO$_{\mathbb{R}}$-Mtsls1 is competitive with the state of the art. We also conducted a second comparison that included 21 extra functions from the special session on continuous optimization of the IEEE 2005 Congress on Evolutionary Computation. From this additional comparison we can conclude that IACO$_{\mathbb{R}}$-Mtsls1 remains very competitive. It mainly shows slightly worse results than G-CMA-ES on functions that are rotated w.r.t. the usual coordinate system. In fact, this is maybe not surprising as G-CMA-ES is the only algorithm of the 20 compared ones that performs very well on these rotated functions. In further work we may test ACO$_{\mathbb{R}}$ in the version that includes the mechanism for adjusting for rotated functions [83] to check whether these potential improvements transfer to IACO$_{\mathbb{R}}$-Mtsls1. Nevertheless, the very good performance of IACO$_{\mathbb{R}}$-Mtsls1 on most of the Soft Computing benchmark functions is a clear indication of the high potential hybrid ACO algorithms have for this problem domain. In fact, IACO$_{\mathbb{R}}$-Mtsls1 is clearly competitive with state-of-the-art continuous optimizers.

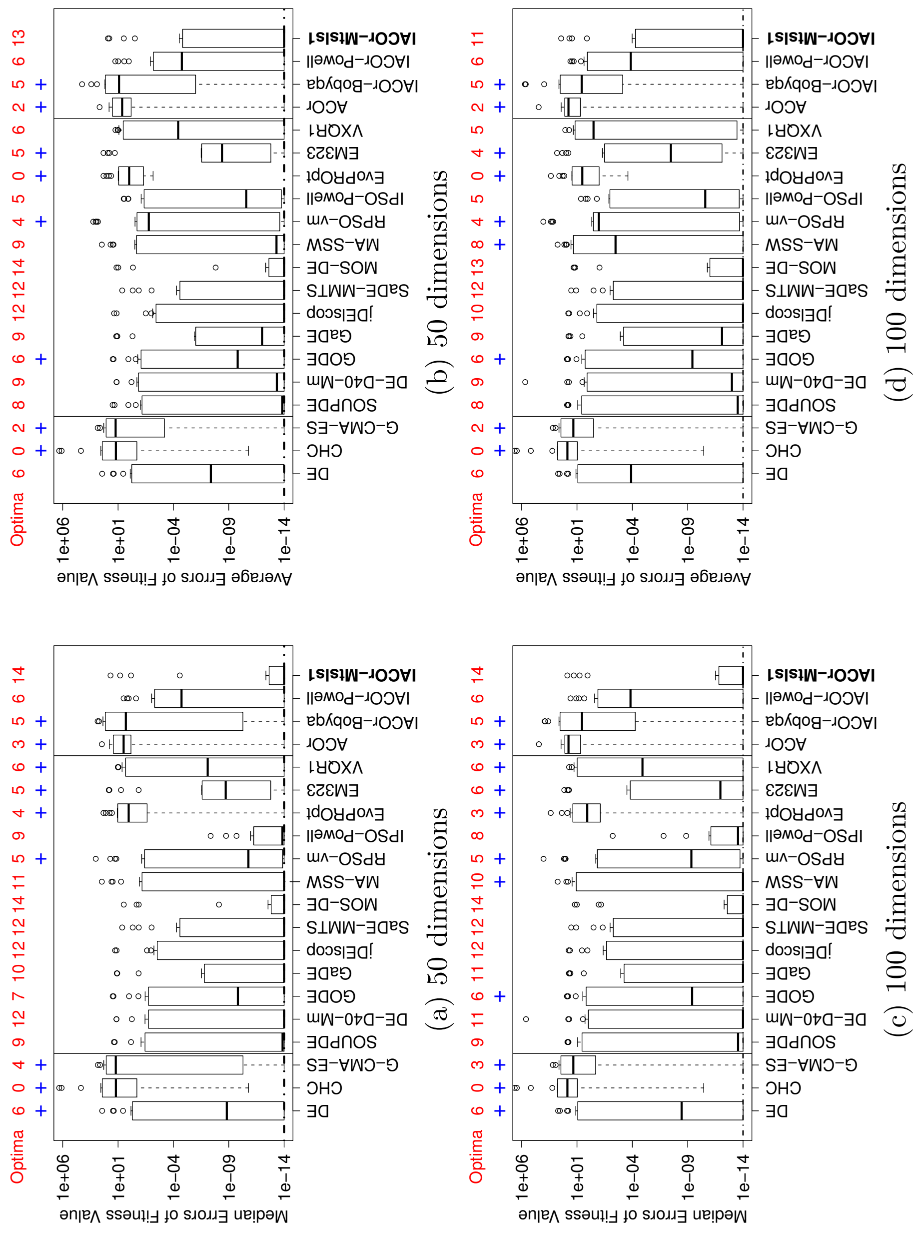

Figure 3.1: The box-plots show the distribution of the median (left) and average (right) errors obtained on the 19 SOCO benchmark functions of 50 (top) and 100 (bottom) dimensions. The results obtained with the three reference algorithms in SOCO are shown on the left part of each plot. The results of 13 algorithms published in SOCO are shown in the middle part of each plot. The results obtained with $\mathrm{ACO}_{\mathbb{R}}$, $\mathrm{IACO}_{\mathbb{R}}$-BOBYQA, $\mathrm{IACO}_{\mathbb{R}}$-Powell, and $\mathrm{IACO}_{\mathbb{R}}$-Mtsls1 are shown on the right part of each plot. The line at the bottom of each plot represents the 0-threshold (10^{-14}). A + symbol on top of a box-plot denotes a statistically significant difference at the 0.05 α-level detected with a Wilcoxon test between the results obtained with the indicated algorithm and those obtained with $\mathrm{IACO}_{\mathbb{R}}$-Mtsls1. The absence of a symbol means that the difference is not significant with $\mathrm{IACO}_{\mathbb{R}}$-Mtsls1. The numbers on top of a box-plot denotes the number of optima found by the corresponding algorithm.

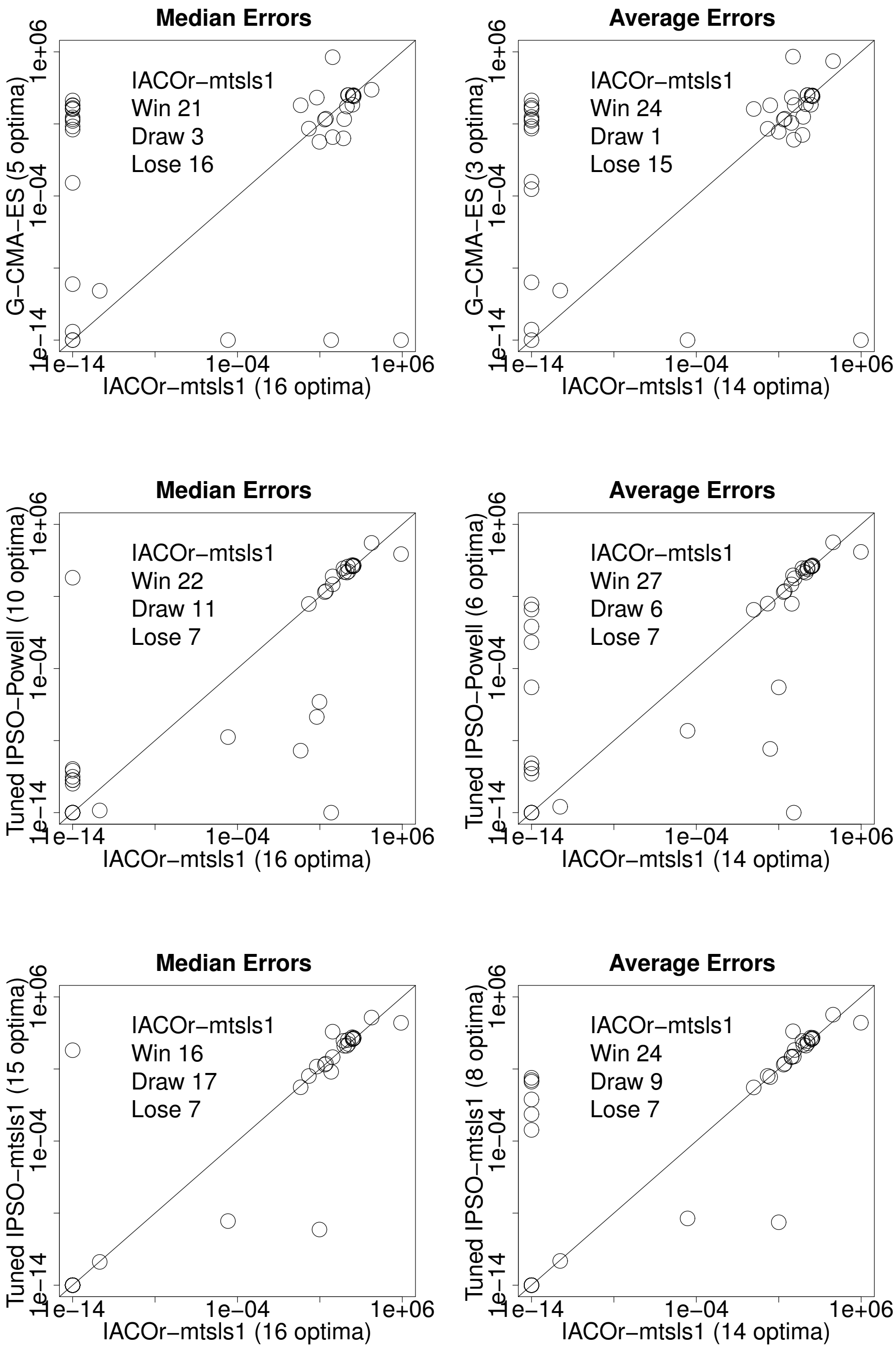

Figure 3.2: The correlation plot between IACO$_{\mathbb{R}}$-Mtsls1 and G-CMA-ES, IPSO-Powell and IPSO-Mtsls1 over 40 functions. Each point represents a function. The points on the left part of correlation plot illustrate that on those represented functions, IACO$_{\mathbb{R}}$-Mtsls1 obtains better results than the other algorithm.

Table 3.3: The comparison is conducted based on median and average errors of objective value and the results of IACO$_{\mathbb{R}}$-Mtsls1 are presented in form of (win, draw, lose), respectively. The tested 40 functions were divided into different properties for details. The two-sided p-values of Wilcoxon matched-pairs signed-rank test of IACO$_{\mathbb{R}}$-Mtsls1 at a 0.05 α-level with other algorithms are also presented

Median Errors

Properties of Functions	IACO$_{\mathbb{R}}$-Mtsls1 vs G-CMA-ES	IACO$_{\mathbb{R}}$-Mtsls1 vs IPSO-Powell	IACO$_{\mathbb{R}}$-Mtsls1 vs IPSO-Mtsls1
Separable	(3, 1, 0)	(0, 4, 0)	(0, 4, 0)
Non-Separable	(18, 2, 16)	(22, 7, 7)	(16, 13, 7)
Non-Separable (Non-Hybrid)	(7, 2, 8)	(6, 6, 5)	(6, 6, 5)
Non-Separable (Hybrid)	(11, 0, 8)	(16, 1, 2)	(10, 7, 2)
Unimodal	(6, 1, 3)	(1, 5, 4)	(1, 5, 4)
Multimodal	(15, 2, 13)	(21, 6, 3)	(15, 12, 3)
Non-rotated	(16, 2, 6)	(10, 8, 6)	(10, 8, 6)
Rotated	(5, 1, 10)	(12, 3, 1)	(12, 3, 1)
SOCO	(15, 2, 2)	(6, 8, 5)	(1, 14, 4)
CEC 2005	(6, 1, 14)	(16, 3, 2)	(15, 3, 3)
In total	(21, 3, 16)	(22, 11, 7)	(16, 17, 7)
p-value	8.33E−01	6.03E−03	1.32E−02

Average Errors

Properties of Functions	IACO$_{\mathbb{R}}$-Mtsls1 vs G-CMA-ES	IACO$_{\mathbb{R}}$-Mtsls1 vs IPSO-Powell	IACO$_{\mathbb{R}}$-Mtsls1 vs IPSO-Mtsls1
Separable	(3, 1, 0)	(1, 3, 0)	(1, 3, 0)
Non-Separable	(21, 0, 15)	(26, 3, 7)	(23, 6, 7)
Non-Separable (Non-Hybrid)	(10, 0, 7)	(9, 3, 5)	(8, 4, 5)
Non-Separable (Hybrid)	(11, 0, 8)	(17, 0, 2)	(15, 2, 2)
Unimodal	(6, 1, 3)	(4, 2, 4)	(2, 4, 4)
Multimodal	(18, 0, 12)	(23, 4, 3)	(22, 5, 3)
Non-rotated	(20, 1, 3)	(13, 5, 6)	(11, 7, 6)
Rotated	(4, 0, 12)	(14, 1, 1)	(13, 2, 1)
SOCO	(16, 1, 2)	(10, 4, 5)	(8, 7, 4)
CEC 2005	(8, 0, 13)	(17, 2, 2)	(16, 2, 3)
In total	(24, 1, 15)	(27, 6, 7)	(24, 9, 7)
p-value	4.22E−01	1.86E−03	1.66E−03

Table 3.4: The median and average errors of objective function values obtained with G-CMA-ES, IPSO-Powell, IPSO-Mtsls1, and IACO$_{\mathbb{R}}$-Mtsls1 on 40 functions with $D = 50$. The lowest values were highlighted in **boldface**. The values below 10^{-14} are approximated to 0. The results of f_{cec1}, f_{cec2}, f_{cec6}, f_{cec9} are not presented to avoid repeated test on the similar functions such as f_{soco1}, f_{soco3}, f_{soco4}, f_{soco8}. At the bottom of the table, we report the number of times an algorithm found the lowest error.

Median errors					Mean errors				
Function	G-CMA-ES	IPSO-Powell	IPSO-Mtsls1	IACO$_{\mathbb{R}}$-Mtsls1	Function	G-CMA-ES	IPSO-Powell	IPSO-Mtsls1	IACO$_{\mathbb{R}}$-Mtsls1
f_{soco1}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco1}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{soco2}	2.64E−11	**1.42E−14**	4.12E−13	4.41E−13	f_{soco2}	2.75E−11	**2.56E−14**	4.80E−13	5.50E−13
f_{soco3}	**0.00E+00**	**0.00E+00**	6.38E+00	4.83E+01	f_{soco3}	7.97E−01	**0.00E+00**	7.29E+01	8.17E+01
f_{soco4}	1.08E+02	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco4}	1.05E+02	**0.00E+00**	1.31E+00	**0.00E+00**
f_{soco5}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco5}	2.96E−04	6.72E−03	5.92E−04	**0.00E+00**
f_{soco6}	2.11E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco6}	2.09E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{soco7}	7.67E−11	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco7}	1.01E−10	4.98E−12	**0.00E+00**	**0.00E+00**
f_{soco8}	**0.00E+00**	1.75E−09	2.80E−10	2.66E−05	f_{soco8}	**0.00E+00**	4.78E−09	4.29E−10	2.94E−05
f_{soco9}	1.61E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco9}	1.66E+01	4.95E−06	**0.00E+00**	**0.00E+00**
f_{soco10}	6.71E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco10}	6.81E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{soco11}	2.83E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{soco11}	3.01E+01	8.19E−02	7.74E−02	**0.00E+00**
f_{soco12}	1.87E+02	1.02E−12	**0.00E+00**	**0.00E+00**	f_{soco12}	1.88E+02	1.17E−11	7.27E−03	**0.00E+00**
f_{soco13}	1.97E+02	**2.00E−10**	5.39E−01	6.79E−01	f_{soco13}	1.97E+02	**2.65E−10**	2.75E+00	3.03E+00
f_{soco14}	1.05E+02	1.77E−12	**0.00E+00**	**0.00E+00**	f_{soco14}	1.09E+02	1.18E+00	5.26E−01	**3.04E−01**
f_{soco15}	8.12E−04	1.07E−11	**0.00E+00**	**0.00E+00**	f_{soco15}	9.79E−04	2.62E−11	**0.00E+00**	**0.00E+00**
f_{soco16}	4.22E+02	3.08E−12	**0.00E+00**	**0.00E+00**	f_{soco16}	4.27E+02	2.80E+00	2.46E+00	**0.00E+00**
f_{soco17}	6.71E+02	**4.35E−08**	1.47E+01	6.50E+00	f_{soco17}	6.89E+02	**3.10E+00**	7.27E+01	6.19E+01
f_{soco18}	1.27E+02	8.06E−12	**0.00E+00**	**0.00E+00**	f_{soco18}	1.31E+02	1.24E+00	1.68E+00	**0.00E+00**
f_{soco19}	4.03E+00	1.83E−12	**0.00E+00**	**0.00E+00**	f_{soco19}	4.76E+00	1.19E−11	**0.00E+00**	**0.00E+00**
f_{cec3}	**0.00E+00**	8.72E+03	1.59E+04	8.40E+05	f_{cec3}	**0.00E+00**	1.24E+04	1.62E+04	9.66E+05
f_{cec4}	4.27E+05	2.45E+02	3.88E+03	**5.93E+01**	f_{cec4}	4.68E+05	2.90E+02	4.13E+03	**7.32E+01**
f_{cec5}	5.70E−01	4.87E−07	**7.28E−11**	9.44E+00	f_{cec5}	2.85E+00	4.92E−06	**2.32E−10**	9.98E+00
f_{cec7}	3.85E−14	**0.00E+00**	**0.00E+00**	**0.00E+00**	f_{cec7}	5.32E−14	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{cec8}	**2.00E+01**	**2.00E+01**	**2.00E+01**	**2.00E+01**	f_{cec8}	2.01E+01	**2.00E+01**	**2.00E+01**	**2.00E+01**
f_{cec10}	**9.97E−01**	8.96E+02	8.92E+02	2.69E+02	f_{cec10}	**1.72E+00**	9.13E+02	8.76E+02	2.75E+02
f_{cec11}	**1.21E+00**	6.90E+01	6.64E+01	5.97E+01	f_{cec11}	**1.17E+01**	6.82E+01	6.63E+01	5.90E+01
f_{cec12}	**2.36E+03**	5.19E+04	3.68E+04	1.37E+04	f_{cec12}	2.27E+05	5.68E+04	5.86E+04	**1.98E+04**
f_{cec13}	4.71E+00	3.02E+00	3.24E+00	**2.14E+00**	f_{cec13}	4.59E+00	3.18E+00	3.32E+00	**2.13E+00**
f_{cec14}	**2.30E+01**	2.35E+01	2.36E+01	2.33E+01	f_{cec14}	**2.29E+01**	2.34E+01	2.35E+01	2.31E+01
f_{cec15}	2.00E+02	2.00E+02	2.00E+02	**0.00E+00**	f_{cec15}	2.04E+02	1.82E+02	2.06E+02	**9.20E+01**
f_{cec16}	**2.15E+01**	4.97E+02	4.10E+02	3.00E+02	f_{cec16}	**3.09E+01**	5.22E+02	4.80E+02	3.06E+02
f_{cec17}	**1.61E+02**	4.54E+02	4.11E+02	4.37E+02	f_{cec17}	**2.34E+02**	4.46E+02	4.17E+02	4.43E+02
f_{cec18}	**9.13E+02**	1.22E+03	1.21E+03	9.84E+02	f_{cec18}	**9.13E+02**	1.18E+03	1.19E+03	9.99E+02
f_{cec19}	**9.12E+02**	1.23E+03	1.19E+03	9.93E+02	f_{cec19}	**9.12E+02**	1.22E+03	1.18E+03	1.01E+03
f_{cec20}	**9.12E+02**	1.22E+03	1.19E+03	9.93E+02	f_{cec20}	**9.12E+02**	1.20E+03	1.18E+03	9.89E+02
f_{cec21}	1.00E+03	1.19E+03	1.03E+03	**5.00E+02**	f_{cec21}	1.00E+03	9.86E+02	8.59E+02	**5.53E+02**
f_{cec22}	**8.03E+02**	1.43E+03	1.45E+03	1.13E+03	f_{cec22}	**8.05E+02**	1.45E+03	1.47E+03	1.14E+03
f_{cec23}	1.01E+03	**5.39E+02**	**5.39E+02**	**5.39E+02**	f_{cec23}	1.01E+03	7.66E+02	6.13E+02	**5.67E+02**
f_{cec24}	**9.86E+02**	1.31E+03	1.30E+03	1.11E+03	f_{cec24}	**9.55E+02**	1.29E+03	1.30E+03	1.10E+03
f_{cec25}	**2.15E+02**	1.50E+03	1.59E+03	9.38E+02	f_{cec25}	**2.15E+02**	1.18E+03	1.50E+03	8.89E+02
# of best	18	15	18	21	# of best	14	10	10	22

Chapter 4

Ant Colony Optimization for Mixed Variable Problems

Recently, many real world problems are modeled using a mixed types of decision variables. A common example is a mixture of discrete variables and continuous variables. The former usually involve ordering characteristics, categorical characteristic or both of them. Due to the practical relevance of such problems, many mixed-variable optimization algorithms have been proposed, mainly based on Genetic Algorithms [38], Differential Evolution [85], Particle Swarm Optimization [51] and Pattern Search [92]. In many cases, the discrete variables are tackled as ordered through a *continuous relaxation approach* [23, 40, 55, 56, 79, 94] based on continuous optimization algorithms. In many other cases, the discrete variables are tackled as categorical through a *native mixed-variable optimization approach* [6, 22, 74] that simultaneous and directly handles both discrete and continuous variables without relaxation. However, It is mentioned that the available approaches are indifferent with either categorical or ordering characteristics of discrete variables. Therefore, there is lack of a generic algorithm which allows to declare each variable of the considered problem as continuous, ordered discrete or categorical discrete.

While ant colony optimization (ACO) was originally introduced to solve discrete optimization problems [24, 25, 87], its adaptation to solve continuous optimization problems enjoys an increasing attention [10, 32, 67] as also discussed in the previous chapter. However, few ACO extensions are applied to mixed-variable optimization problems.

In this chapter, we present $\text{ACO}_{\mathbf{MV}}$, an $\text{ACO}_{\mathbb{R}}$ extension for mixed-variable optimization problems. $\text{ACO}_{\mathbf{MV}}$ integrates a component of a continuous relaxation approach ($\text{ACO}_{\mathbf{MV}}$-o) and a component of a native mixed-variable optimization approach ($\text{ACO}_{\mathbf{MV}}$-c), as well as $\text{ACO}_{\mathbb{R}}$ and allows to declare each variable of the mixed variable optimization problems as continuous, ordered discrete or categorical discrete. We also propose a new set of artificial mixed-variable benchmark functions and their constructive methods, thereby providing a flexibly controlled environment for investigating the performance and training parameters of mixed-variable optimization algorithms. We automatically tune the parameters of $\mathrm{ACO}_{\mathbf{MV}}$ by the iterated F-race method [9, 13]. Then, we not only evaluate the performance of $\mathrm{ACO}_{\mathbf{MV}}$ on benchmark functions, but also compare the performance of $\mathrm{ACO}_{\mathbf{MV}}$ on 4 classes of 8 mixed-variables engineering optimization problems with the results from literature. $\mathrm{ACO}_{\mathbf{MV}}$ has efficiently found all the best-so-far solution including two new best solution. $\mathrm{ACO}_{\mathbf{MV}}$ obtains 100% success rate in 7 problems. In 5 of those 7 problems, $\mathrm{ACO}_{\mathbf{MV}}$ requires the smallest number of function evaluations. To sum up, $\mathrm{ACO}_{\mathbf{MV}}$ has the best performance on mixed-variables engineering optimization problems from the literature.

4.1 Mixed-variable Optimization Problems

A model for a mixed-variable optimization problem (MVOP) may be formally defined as follows:

Definition *A model $R = (\mathbf{S}, \mathbf{\Omega}, f)$ of a MVOP consists of*

- *a search space $\mathbf{S}$ defined over a finite set of both discrete and continuous decision variables and a set $\mathbf{\Omega}$ of constraints among the variables;*
- *an objective function $f : \mathbf{S} \to \mathbb{R}_0^+$ to be minimized.*

The search space $\mathbf{S}$ is defined as follows: Given is a set of $n = d + r$ variables $X_i, i = 1, \ldots, n$, of which d are discrete with values $v_i^j \in \mathbf{D}_i = \{v_i^1, \ldots, v_i^{|\mathbf{D}_i|}\}$, and r are continuous with possible values $v_i \in \mathbf{D}_i \subseteq \mathbb{R}$. Specifically, the discrete search space is expanded to be defined as a set of $d = o + c$ variables, of which o are ordered and c are categorical discrete variables, respectively. A solution $s \in \mathbf{S}$ is a complete assignment in which each decision variable has a value assigned. A solution that satisfies all constraints in the set $\mathbf{\Omega}$ is a feasible solution of the given MVOP. If the set $\mathbf{\Omega}$ is empty, R is called an unconstrained problem model, otherwise it is said to be constrained. A solution $s^* \subseteq \mathbf{S}$ is called a global optimum if and only if: $f(s^*) \leq f(s)\ \forall_{s \in \mathbf{S}}$. The set of all globally optimal solutions is denoted by $\mathbf{S}^* \subseteq \mathbf{S}$. Solving a MVOP requires finding at least one $s^* \subseteq \mathbf{S}^*$.

The methods proposed in the literature to tackle MVOPs may be divided into three groups.

The first group is based on a *two-partition approach*, in which the mixed variables are decomposed into two partitions, one involving the continuous variables and the other involving the discrete variables. Variables of one partition are optimized separately for fixed values of the variable of the other partition [78]. The approach usually leads to a large number of objective function evaluations [84] and the dependency of variables may lead to a sub-optimal solution. More promising are the other two groups. The second group is a *continuous relaxation approach.* Discrete variables are relaxed to continuous variables, but are repaired when evaluating the objective function. The repair mechanism is used to return a discrete variable in each iteration. The simplest repair mechanisms are by truncation and rounding [40, 55]. The performance depends on the continuous solvers and the repair mechanisms. The third group is a *native mixed-variable optimization approach* that simultaneously and directly handles both discrete and continuous variables without relaxation. It is indifferent to the ordering character of the discrete variables. Genetic adaptive search, pattern search, and mixed bayesian optimization are among the approaches that have been proposed in [6, 22, 74].

A particular class of MVOPs is known as mixed variable programming (MVP) problems [6]. They are characterized by a combination of continuous and categorical variables. The latter are discrete variables that take their values from a set of categories [4]. Categorical variables often identify non-numeric elements of an unordered set (colors, shapes or type of materials) and characterize the structure of problems [65]. Therefore, the discreteness of categorical variables must be satisfied at every iteration when considering potential iterative solution approaches [2].

However, MVP problems are not considered about the ordering nature of discrete variables, and the solvers for MVP problems are almost based on a native mixed-variable optimization approach. Therefore, Those solvers may not efficiently handle highly ordered variables owing to the lack of continuous relaxations.

In another aspect, [1] claims modeling without categorical variables so that continuous relaxations may be used to handle categorical variables. But the performance of continuous relaxations may decline with an increasing number of categories. Therefore, a possible more rigorous way of classifying MVOPs is to consider whether the discrete variables are ordered or categorical ones, since they are both important characters for discrete variables.

Whereas, researchers often take one specific group of approaches to develop mixed-variable optimization algorithms and test on MVOPs with one specific type of discrete variables, finally obtain reasonable good results, rather than investigating those algorithms on MVOPs with other types of discrete variables. Therefore, there is lack of rigorous comparisons between continuous relaxation approach and native mixed-variable optimization approach, let alone taking the advantage of the strategies of the both approaches to improve algorithms performance on more general and various MVOPs. However, in our study, we have done those work in Section 4.4 and Section 4.6.

4.2 ACO$_{\mathbf{MV}}$ Heuristics for Mixed-Variable Optimization Problems

We start by describing the ACO$_{\mathbf{MV}}$ heuristic framework, and then, we describe the probabilistic solution construction for continuous variables, ordered discrete variables and categorical variables, respectively.

4.2.1 ACO$_{\mathbf{MV}}$ framework

The basic flow of the ACO$_{\mathbf{MV}}$ algorithm is as follows. As a first step, the solution archive is initialized. Then, at each iteration a number of solutions is probabilistically constructed by the ants. These solutions may be improved by any improvement mechanism (for example, local search or gradient techniques). Finally, the solution archive is updated with the generated solutions. In the following we outline the archive structure, the initialization and the update of the archive in more details.

ACO$_{\mathbf{MV}}$ keeps a history of its search process by storing solutions in a *solution archive* T of dimension $|T| = k$. Given an n-dimensional MVOP and k solutions, ACO$_{\mathbf{MV}}$ stores the values of the solutions' n variables and the solutions' objective function values in T. The value of the i-th variable of the j-th solution is in the following denoted by s_j^i. Figure 4.1 shows the structure of the solution archive. It is divided into three groups of columns, one for categorical variables, one for ordered discrete variables and one for continuous variables. ACO$_{\mathbf{MV}}$-c and ACO$_{\mathbf{MV}}$-o handle categorical variables and ordered discrete variables, respectively, while ACO$_{\mathbb{R}}$ handles continuous variables.

Before the start of the algorithm, the archive is initialized with k random solutions. At each algorithm iteration, first, a set of m solutions is generated by the ants and added to those in T. From this set of $k + m$ solutions, the m worst ones are removed. The remaining k solutions are sorted according to their quality (i.e., the value of the objective function) and stored in the new T. In this way, the search process is biased towards the best solutions found during the search. The solutions in the archive are always kept sorted based on their quality, so that the best solution is on top. An outline of the ACO$_{\mathbf{MV}}$ algorithm is given in Algorithm 4.

4.2.2 Probabilistic Solution Construction for Continuous Variables

Continuous variables are handled by ACO$_{\mathbb{R}}$ [83], which has been further explained in Chapter 2.1.

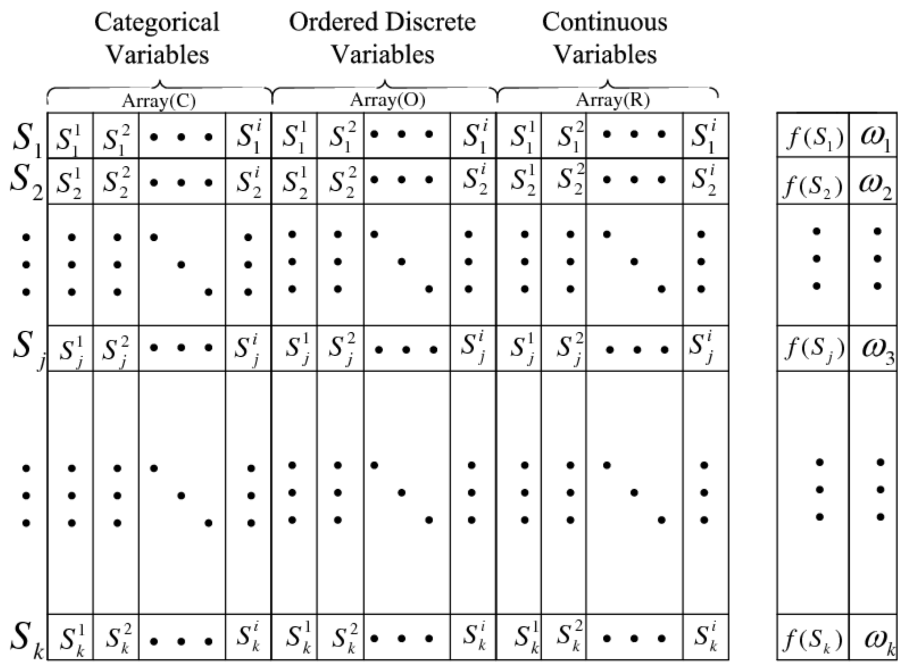

Figure 4.1: The structure of the solution archive of ACO$_{\mathbf{MV}}$. The solutions in the archive are sorted according to their quality, i.e., the value of the objective function $f(s_i)$), hence, the position of a solution in the archive always corresponds to its rank.

4.2.3 Probabilistic Solution Construction for Ordered Discrete Variables

If *ordered discrete variables* are defined, a component of the continuous relaxation approach, ACO$_{\mathbf{MV}}$-o, is used. The natural ordering of the values for these variables may have little to do with their actual numerical values (and they may even not have numerical values, e.g., $x \in \{\text{small}, \text{big}, \text{huge}\}$). Hence, instead of operating on the actual values of the ordered discrete variables, ACO$_{\mathbf{MV}}$-o operates on their indexes. The values of the indexes for the new solutions are generated as real numbers, as it is the case for the continuous variables. However, before the objective function is evaluated, the continuous values are rounded to the nearest valid index, and the value at that index is then used for the objective function evaluation. At the algorithm level, ordered discrete variables are transformed into continuous variables for probabilistically constructing solution.

Algorithm 4 Outline of ACO$_{\mathbf{MV}}$

Initialize decision variables
  Categorical varables $\rightarrow array(C)$
  Ordering discrete varables $\rightarrow array(O)$
  Continuous varables $\rightarrow array(R)$
// Initialize pheromones
Initialize solution archive(T) of size K
while termination criterion not satisfied **do**
  // ConstructAntSolution
  for $n = 1$ to $Nants$ **do**
    // ConstructSolution($S_1 \cdots S_{Nants}$)
    Probabilistic Solution Construction for ACO$_{\mathbf{MV}}$-c
    Probabilistic Solution Construction for ACO$_{\mathbf{MV}}$-o
    Probabilistic Solution Construction for ACO$_{\mathbb{R}}$
  end for
  Tnew= $First_k \Leftarrow Rank(S(T) \cup S_1 \cdots S_{Nants})$
  // Update pheromones
  Update solution achive(T)
end while

4.2.4 Probabilistic Solution Construction for Categorical Variables

While ordered discrete variables are relaxed and treated in the original ACO$_{\mathbb{R}}$, *categorical variables* are treated differently in a component of the native discrete optimization approach, ACO$_{\mathbf{MV}}$-c, as for this type of variables there is no pre-defined ordering. The pheromone representation (i.e., the *solution archive*) as well as the general flow of ACO$_{\mathbf{MV}}$ does not change. Hence, we focus here on presenting how the discrete variables are handled without the ordered information in the domain. The values for these variables are generated with a different method—one that is closer to the regular combinatorial ACO.

In standard ACO (see [26]), solutions are constructed from solution components using a probabilistic rule based on the pheromone values. Differently, in ACO$_{\mathbf{MV}}$ there are no static pheromone values, but a *solution archive*. As in standard ACO, in ACO$_{\mathbf{MV}}$-c, the construction of solutions for categorical variables is done by choosing the components, that are, the values for each of the categorical decision variables. However, since the pheromone model of standard ACO are replaced by the solution archive, the probabilistic solution construction rule is modified as follows.

Similarly to the case of continuous variables, each ant constructs the categorical discrete part of the solution incrementally. For each categorical variable i, each ant chooses probabilistically one of c^i available values $v_l^i \in$

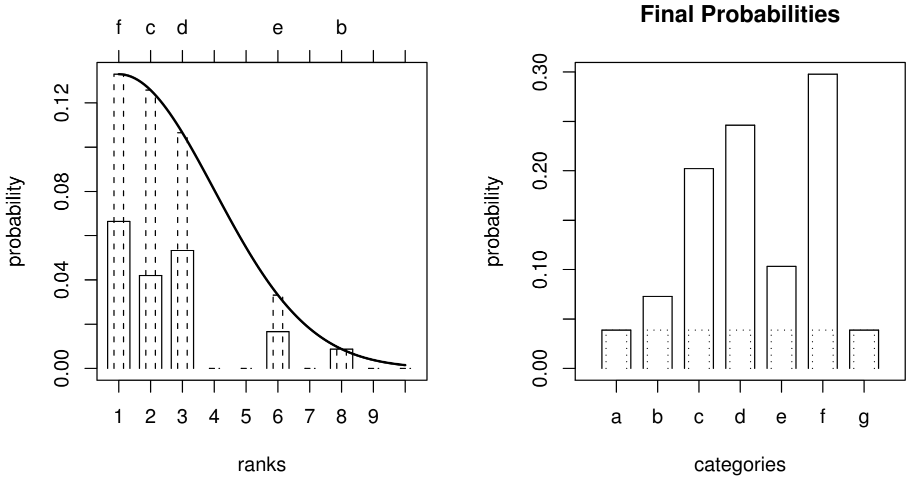

Figure 4.2: Calculating probabilities of choosing different categorical values for a given decision variable. First, the initial probabilities are generated using a normal distribution and based on the best ranked solution that uses given value (left plot, dashed bars). Then, they are divided by the number of solutions using this value (left plot, solid bars), and finally a fixed value is added (right plot, dotted bars) in order to increase the probability of choosing those values, which are currently not used. The final probabilities are presented on the right plot, as solid bars.

$\mathbf{D}_i = \{v_1^i, ..., v_{c^i}^i\}$. The probability of choosing the l-th value is given by:

$$o_l^i = \frac{w_l}{\sum_{r=1}^{c} w_r}, \tag{4.1}$$

where w_l is the weight associated with the l-th available value. It is calculated based on the weights ω and some additional parameters:

$$w_l = \frac{\omega_{j_l}}{u_l^i} + \frac{q}{\eta}. \tag{4.2}$$

The final weight w_l is hence a sum of two components. The weight ω_{j_l} is calculated according to Equation 2.2, where the j_l is the index of the highest quality solution that uses value v_l^i for the i-th categorical variable. In turn, u_l^i is the number of solutions using value v_l^i for the i-th categorical variable in the archive. Therefore, the more *popular* the value v_l^i is, the lower is its final weight.

The second component is a fixed value (i.e., it does not depend on the value v_l^i chosen): η is the number of values from the c^i available ones that are unused by the solutions in the archive, and q is the same parameter of the algorithm that was used in Equation 2.2.

The graphical representation of how the first component $\frac{\omega_{j_l}}{u_l^i}$ is calculated is presented on the left plot of Figure 4.2. The dashed bars indicate the values of the weights ω_{j_l} obtained for the best solutions using the available values. [1] The solid bars represent the weights ω_{j_l} divided by the respective number of solutions u_l^i that use values v_l^i. It is shown for the available set of categorical values used, $v_l^i \in \{\mathrm{a, b, c, d, e, f, g}\}$ in this example.

Some of the available categorical values v_l may be unused for a given i-th decision variable in all the solutions in the archive. Hence, their initial weight is zero. In order to enhance exploration and to prevent premature convergence, in such a case, the final weights w are further modified by adding to all of them the second component. Its value depends on the parameter q and on the number of unused categorical values η_i, as shown in Equation 4.2.

The right plot in Figure 4.2 presents the normalized final probabilities for an example in which the solution archive has size $k = 10$, and where the set of categorical values is $\{\mathrm{a, b, c, d, e, f, g}\}$, with values $\{\mathrm{a}\}$ and $\{\mathrm{g}\}$ unused by the current decision variable. The dotted bars show the value of q/η added to all the solutions, and the solid bars show the final resulting probabilities associated with each of the available categories. These probabilities are then used to generate the value of the i-th decision variable for the new solutions.

4.2.5 Auxiliary Explanations of $\mathrm{ACO}_{\mathbf{MV}}$

The following are some auxiliary explanations of $\mathrm{ACO}_{\mathbf{MV}}$. ACO algorithms in general do not exploit correlation information between different decision variables (or components). In $\mathrm{ACO}_{\mathbf{MV}}$, due to the specific way the pheromone is represented (i.e., as the solution archive), it is in fact possible to take into account the correlation between the decision variables. A non-deterministic adaptive method is presented in [83], which will take effect on rotated benchmark functions proposed in Section 4.3, and also handle variable dependency of engineering problem in Section 4.6.

For simplification of $\mathrm{ACO}_{\mathbf{MV}}$, The uniform random sampling in the range of decision variables is used for initial solution archive. For fighting stagnation, a simple restart strategy consists in restarting the algorithm but keeping the best-so-far solution in archive. The restart criterion is the number of iterations of ants updating the archive with a relative solution improvement lower than a certain threshold ε. $\mathrm{ACO}_{\mathbf{MV}}$ is implemented in C++.

[1] If a given value is not used, the associated index is indefinite, and thus its initial weight is zero.

4.3 Artificial Mixed-variable Benchmark Functions

The mixed-variable benchmark problems found in the literature often originate from the mechanical engineering field, which can not be easily parametrized and flexibly manipulated for investigating the performance of mixed-variable optimization algorithms.

In this section, we propose a set of new, artificial mixed-variable benchmark functions for a sufficiently *controlled environment* for the investigation of the performance and the automatic parameter tuning of algorithms. Our proposed artificial mixed-variable benchmark functions are defined in Table 4.1. The expressions of objective functions originate from some typical continuous functions in IEEE CEC 2005. The decision variables consist of discrete and continuous variables. n is the number of dimensions including discrete variables and continuous variables and $\mathbf{M}$ is a random, normalized n-dimensional rotation matrix. The continuous variables' global optima are shifted to avoid a bias of population based methods towards the center of the search space [34]. It allows 3 settings for discrete variables, one involving ordered discrete variables, one involving categorical variables and one involving mixed ordered discrete and categorical variables.

In order to make it easier to understand and visualize the benchmark functions, we use the two dimensional, not shifted, randomly rotated Ellipsoid mixed-variable functions as an example to illustrate the principle of how to construct artificial mixed-variable benchmark functions. Equation 4.3 is randomly rotated Ellipsoid continuous function.

$$f_{EL}(\vec{x}) = \sum_{i=1}^{n} (\beta^{\frac{i-1}{n-1}} z_i)^2, \quad \begin{cases} \vec{x} \in (-3,7)^n, \\ \vec{z} = \mathbf{M}\vec{x}, \end{cases} \tag{4.3}$$

In order to transform this continuous function into a mixed-variable one, we have divided the continuous domain of variable $x_1 \in (-3,7)$ into a set of discrete values, $\mathbf{T} = \{\theta_1, \theta_2, ..., \theta_t\} : \theta_i \in (-3,7)$. This results in the following mixed-variable test function:

$$f_{EL_{MV}}(\vec{x}) = z_1^2 + \beta \cdot z_2^2, \quad \begin{cases} x_1 \in \mathbf{T}, \\ x_2 \in (-3,7), \\ \vec{z} = \mathbf{M}\vec{x}. \end{cases} \tag{4.4}$$

The set $\mathbf{T}$ is created by choosing t uniformly spaced values from the original domain $(-3,7)$ in such a way that $\exists_{i=1,...,t}\ \theta_i = 0$. This way, it is always possible to find the optimum value $f_{EL_{MV}}(0,0) = 0$, regardless of the chosen t discrete values.

In the following, we will explain the first two setups of discrete variables to simulate each benchmark function, respectively: (i) with ordered discrete variables, and (ii) with categorical variables. In the first setup, the discrete

Table 4.1: Artificial mixed-variable benchmark functions

The objective functions
$f_{Ellipsoid_{MV}}(\vec{x}) = \sum_{i=1}^{n}(\beta^{\frac{i-1}{n-1}} z_i)^2,$
$f_{Ackley_{MV}}(\vec{x}) = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}(z_i^2)}} - e^{\frac{1}{n}\sum_{i=1}^{n}(\cos(2\pi z_i))} + 20 + e,$
$f_{Rastrigin_{MV}}(\vec{x}) = 10n + \sum_{i=1}^{n}(z_i^2 - 10\cos(2\pi z_i^2)),$
$f_{Rosenbrock_{MV}}(\vec{x}) = \sum_{i=1}^{n-1}[100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2],$
$f_{Sphere_{MV}}(\vec{x}) = \sum_{i=1}^{n} z_i^2,$
$f_{Griewank_{MV}}(\vec{x}) = \frac{1}{4000}\sum_{i=1}^{n} z_i^2 - \prod_{i=1}^{n}\cos(\frac{z_i}{\sqrt{i}}) + 1,$
The definition of mixed variables
$\vec{x}_d \in \mathbf{T}, \mathbf{T} = \{\theta_1, \theta_2, ..., \theta_t\} : \theta_i \in (MinRange, MaxRange)$
$\vec{x}_r \in (MinRange, MaxRange),$
$\vec{x} = \vec{x}_d \oplus \vec{x}_r, \vec{x} \in (MinRange, MaxRange)^n,$
$n = \|d\| + \|r\|,$
$\vec{z} = (\vec{x} - \vec{o})\mathbf{M},$
$\vec{o}_{global\ optima} = [0_1, 0_2, ..., 0_D, o_1, o_2, ..., o_C]:$
The 1st setting for $\vec{x}_d$: $\vec{x}_d$ involves $\vec{x}_{ordered}$
The 2nd setting for $\vec{x}_d$: $\vec{x}_d$ involves $\vec{x}_{categorical}$
The 3rd setting for $\vec{x}_d$: $\vec{x}_d$ involves $\vec{x}_{ordered} \oplus \vec{x}_{categorical}$

intervals for variable x_1 are naturally ordered. Such a setup simulates a problem where the ordering of the discrete variables may be easily defined. The left plot in Figure 4.3 shows how the algorithm sees such a naturally ordered rotated ellipsoid function, with discrete x_1 variable.[2] The test function is presented as a set of points representing different solutions found by the ants and stored in the solution archive. The darker the point, the higher the quality of the solution. In the second setup, the intervals are ordered randomly, that is, for each run of the algorithm a different ordering was generated. This setup allows to investigate how the algorithm performs when the optimum ordering of the intervals is not well defined or unknown. The right plot of Figure 4.3 shows how the algorithm sees such modified problem for a given single random ordering. Therefore, the discrete variables become categorical without natural ordering. Clearly, compared to the natural ordering, the problem appears to be quite different.

The artificial mixed-variable benchmark functions also consist of the

[2]Please note that Figure 4.3 uses the value of $\beta = 5$, as it is clearer for visualization. This simply means that the ellipsoid is less flat and more circle-like.

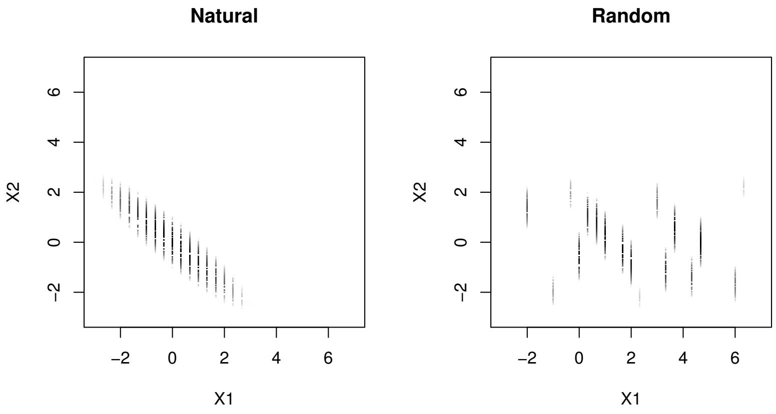

Figure 4.3: Randomly rotated ellipsoid function ($\beta = 5$) with discrete variable $x_1 \in \mathbf{T}, |\mathbf{T}| = t = 30$. The left plot presents the case in which the natural ordering of the intervals is used, while the right one presents the case in which a random ordering is used.

characteristics such as non-separable, ill-conditioned and multi-modal. Nonseparate functions often exhibit intricate dependencies between decision variables. Ill-conditioned functions, like $f_{Rosenbrock_{MV}}$, often lead to premature convergence. Multi-modal functions, like $f_{Ackley_{MV}}$, $f_{Rastrigin_{MV}}$ and $f_{Griewank_{MV}}$, serves to find effectively a search globally in a highly multimodal topography [43]. For example, in the continuous study of [8], we can see PSO performs well on the separable problems. However, on nonseparable problems, PSO exhibits a strong performance decline, and PSO also performs very poorly even on moderately ill-conditioned functions, let alone in mixed-variable optimization cases. Therefore, the proposed artificial mixed-variable benchmark functions are expected to lead a challenge for different mixed-variable optimization algorithms. In anther aspect, the flexible discrete intervals and dimensions of the proposed benchmark functions are not only helpful for investigating the performance scalability of mixed-variable optimization algorithms, but also provide a convenient environment for automatic parameter tuning in mixed-variable optimization solvers generalization, thereby facing unseen real-world complex engineering optimization problems.

4.4 Performance Evaluation of $\mathbf{ACO_{MV}}$-o and $\mathbf{ACO_{MV}}$-c

$\text{ACO}_{\mathbf{MV}}$-o and $\text{ACO}_{\mathbf{MV}}$-c represent a continuous relaxation approach and a native mixed-variable optimization approach on handling discrete variables, respectively. We evaluate the performance of $\text{ACO}_{\mathbf{MV}}$-o and $\text{ACO}_{\mathbf{MV}}$-c on two different setups of mixed-variable benchmark functions proposed in previous Section 4.3. The first setups of benchmark functions involve ordered discrete variables. The second setups of benchmark functions involve categorical variables. The goal of the first setups is to evaluate and compare the performance of $\text{ACO}_{\mathbf{MV}}$-o and $\text{ACO}_{\mathbf{MV}}$-c in the case that the discrete variables are ordered. The objective of the second setups is to evaluate and compare the performance of $\text{ACO}_{\mathbf{MV}}$-o and $\text{ACO}_{\mathbf{MV}}$-c in the case that the discrete variables are categorical. Based on the experimental results, we can find that hybrid of $\text{ACO}_{\mathbf{MV}}$-o and $\text{ACO}_{\mathbf{MV}}$-c in $\text{ACO}_{\mathbf{MV}}$ consist in taking the respective advantage for handling corresponding setup of discrete variables.

4.4.1 Experimental Setup

For both setups of six benchmark functions in previous Section 4.3, we evaluate the performance of $\text{ACO}_{\mathbf{MV}}$-o and $\text{ACO}_{\mathbf{MV}}$-c on a different number t of intervals $t \in \{2, 5, 10, 20, ..., 90, 100, 200, ...900, 1000\}$ and on the dimensions (2, 6 and 10)[3]. It not only shows solution quality on different dimensions, also shows the impact of the interval size on the algorithm performance. For each setup of discrete variables, we conduct 18 groups of experiments for comparison in total, involving 6 different benchmark functions with 3 different dimensions. In every group of experiment, in order to ensure a fair comparison of $\text{ACO}_{\mathbf{MV}}$-o and $\text{ACO}_{\mathbf{MV}}$-c, we tuned their parameters using the same tuning procedure: the Iterated F-race method [9,13] which combines F-Race [11,12] with a process capable of generating promising candidate configurations. In the training set of off-line tuning, we use 200 instances of same benchmark function with the same dimension, but involving ordered discrete and categorical variables, random intervals $t \in \{2, 5, 10, 20, ..., 90, 100, 200, ...900, 1000\}$ and random function's coefficients. The tuning budget is set up to 2000 evaluations. In a production phase, we have conducted 21 [4] comparison experiments across the intervals in every group of experiment. In total, we have conducted $378(21 \times 6 \times 3)$[5] times of comparison experiments for each setup of discrete variables. Every time of experiment, we investigate solution quality by 50 independent runs to compare $\text{ACO}_{\mathbf{MV}}$ involving $\text{ACO}_{\mathbf{MV}}$-o and involving $\text{ACO}_{\mathbf{MV}}$-c, without restart mechanism[6]. The pure random search method is included as a baseline for comparison.

[3] In this study, the dimensionality of mixed-variable functions consists in the half dimensional discrete variables and the other half dimensional continuous variables.

[4] 21 intervals $t \in \{2, 5, 10, 20, ..., 90, 100, 200, ...900, 1000\}$

[5] 21 intervals, 6 benchmark functions and 3 different dimensions

[6] It is for the pure comparison of $\text{ACO}_{\mathbf{MV}}$-o and $\text{ACO}_{\mathbf{MV}}$-c. Restart mechanism is included in $\text{ACO}_{\mathbf{MV}}$ for the performance evaluation in later sections

4.4.2 Comparison Results

In the case of first setups involving ordered discrete variables, Wilcoxon rank-sum test with significance level 0.05 is used on each comparison of 378 comparison experiments. Totally, the comparison result of $\mathrm{ACO}_{\mathbf{MV}}$-o and $\mathrm{ACO}_{\mathbf{MV}}$-c is (0.63, 0.35, 0.02), which indicates that $\mathrm{ACO}_{\mathbf{MV}}$-o outperform $\mathrm{ACO}_{\mathbf{MV}}$-c with a probability 0.63, while outperformed by $\mathrm{ACO}_{\mathbf{MV}}$-c with a probability 0.02. Their statistical insignificance is a probability of 0.35. Similarly, the comparison result of $\mathrm{ACO}_{\mathbf{MV}}$-o and random search is (0.98, 0.02, 0). The comparison result of $\mathrm{ACO}_{\mathbf{MV}}$-c and random search are (0.93, 0.07, 0). In the case of second setups involving categorical variables, the comparison result of $\mathrm{ACO}_{\mathbf{MV}}$-o and $\mathrm{ACO}_{\mathbf{MV}}$-c is presented (0.07, 0, 0.93), which indicates that $\mathrm{ACO}_{\mathbf{MV}}$-o outperform $\mathrm{ACO}_{\mathbf{MV}}$-c with a probability 0.07, while outperformed by $\mathrm{ACO}_{\mathbf{MV}}$-c with a probability 0.93. Similarly, the comparison result of $\mathrm{ACO}_{\mathbf{MV}}$-o and random search is (0.78, 0.12, 0.10). The comparison result of $\mathrm{ACO}_{\mathbf{MV}}$-c and random search are (0.96, 0.04, 0).

We conclude statistically that, in $\mathrm{ACO}_{\mathbf{MV}}$, $\mathrm{ACO}_{\mathbf{MV}}$-o is more efficient than $\mathrm{ACO}_{\mathbf{MV}}$-c in the case that discrete variables of mixed-variable problems are ordered, while $\mathrm{ACO}_{\mathbf{MV}}$-c is more efficient than $\mathrm{ACO}_{\mathbf{MV}}$-o in the case that discrete variables of mixed-variable problems are categorical variables, for which no obvious ordering exists. Meanwhile, The experimental results illustrate the advantage of hybrid the $\mathrm{ACO}_{\mathbf{MV}}$-o and $\mathrm{ACO}_{\mathbf{MV}}$-c for handling corresponding class of discrete variables. Figures 4.4 and 4.5 are some examples in the comparisons.

As seen from those figures, the mean performance of $\mathrm{ACO}_{\mathbf{MV}}$-c does not differ from the two different setups of the benchmark functions. The mean performance of $\mathrm{ACO}_{\mathbf{MV}}$-c decreases slightly with the increase of the number of intervals. This shows that the ordering of the intervals should not matter for a native mixed-variable optimization approach. Its efficiency depends only on the number of intervals. The more there are intervals, the more difficult it becomes to find the optimal one. However, the mean performance of $\mathrm{ACO}_{\mathbf{MV}}$-o differ greatly from two different setups. There is no obvious trend as the increase of the number of intervals.

4.5 Performance Evaluation of $\mathrm{ACO}_{\mathbf{MV}}$

We automatically tune the parameters of $\mathrm{ACO}_{\mathbf{MV}}$ by Iterated F-Race. Then, we investigate the performance of $\mathrm{ACO}_{\mathbf{MV}}$ on artificial mixed-variable benchmark functions in Section 4.3, as well as the restart mechanism of $\mathrm{ACO}_{\mathbf{MV}}$ on fighting stagnation by analyzing the algorithmsâ qualified run-length distributions (RLDs).

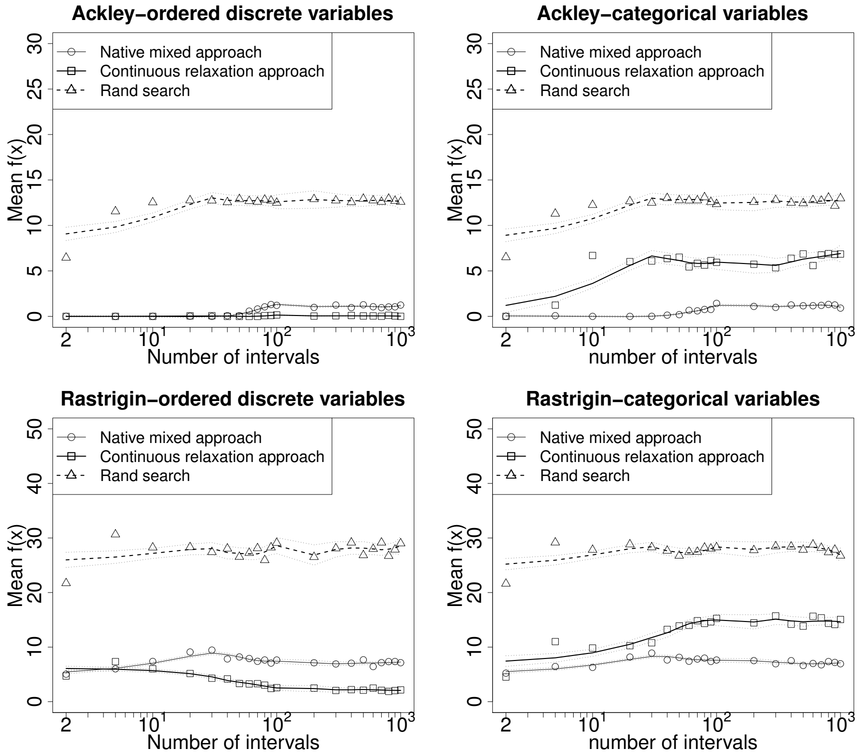

Figure 4.4: The mean value evaluation of $ACO_{\mathbf{MV}}$-o and $ACO_{\mathbf{MV}}$-c on 6 dimensional benchmark functions after 10000 evaluations, with intervals $t \in \{2, 5, 10, 20, ..., 90, 100, 200, ..., 900, 1000\}$

.

4.5.1 Parameter Tuning of $ACO_{\mathbf{MV}}$

A crucial aspect of mixed-variable algorithms' parameter configuration is generalization. Given a set of artificial mixed-variable benchmark functions as training instances, our goal is to find high-performing algorithm parameters that perform well on unseen problems that are not available when deciding on the algorithm parameters [13]. Therefore, we avoid over-tuning by applying Iterated F-Race to artificial mixed-variable benchmark functions rather than the engineering problems , which $ACO_{\mathbf{MV}}$ are tested and compared in Section 4.6. For the generalization of parameters, the instances of training set are designed across six mixed-variable benchmark functions with mixed dimensions(2, 4, 6, 8, 10, 12, 14) [61], involving two setups of benchmark functions,(i) with ordered discrete variables, and (ii) with categorical variables. We use 300 random instances and 5000 budget of experimental evaluations in the automatic tuning procedure. The parameters obtained are in Table 4.2. it is used for performance evaluation of $ACO_{\mathbf{MV}}$ later , and also for real world engineering optimization problems in Section 4.6.

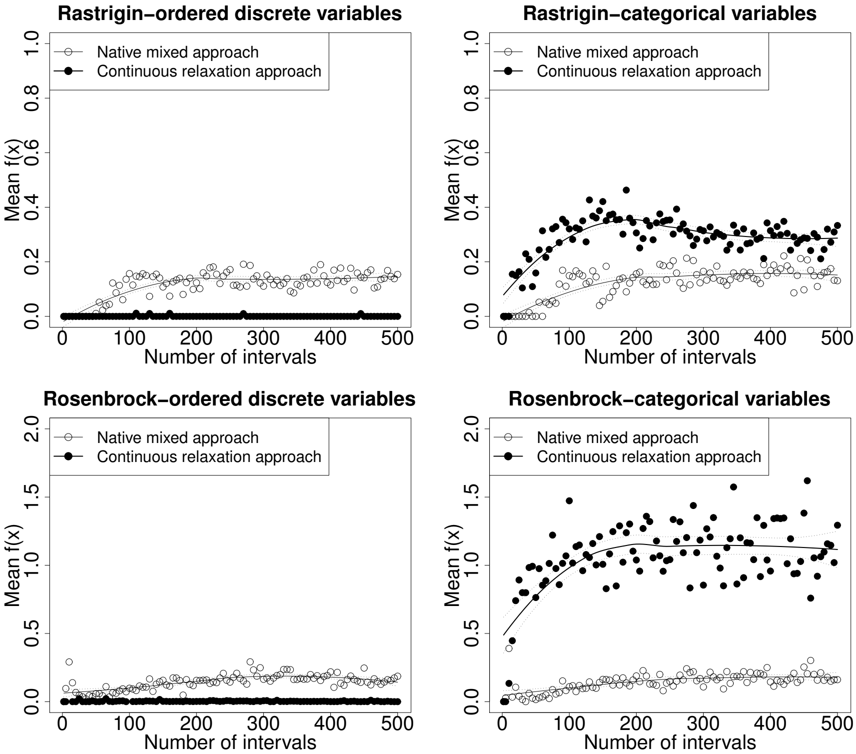

Figure 4.5: The mean value evaluation of ACO_{MV}-o with ACO_{MV}-c on 2 dimensional benchmark functions after 10000 function evaluations, with intervals $t \in \{2, 5, 10, 15, ..., 490, 495, 500\}$

Table 4.2: Summary on the tuned parameters of ACO_{MV}.

Parameter	Symbol	Value
Number of ants	m	5
Speed of convergence	ξ	0.05099
Locality of the search	q	0.6795
Archive size	k	90

4.5.2 The Performance of Fighting Stagnation

Firstly, we evaluate the performance of ACO_{MV} on the two setups of artificial mixed-variable benchmark functions proposed in Section 4.3 with dimensions (2, 6, 10). Table 4.3 shows the experimental results on the discrete variables' intervals $t = 100$. ACO_{MV} solved all 2 dimensional benchmark functions with 100% success rate. ACO_{MV} found the optimal solution of all the 6 dimensional benchmark functions. On the 10 dimensional bench-

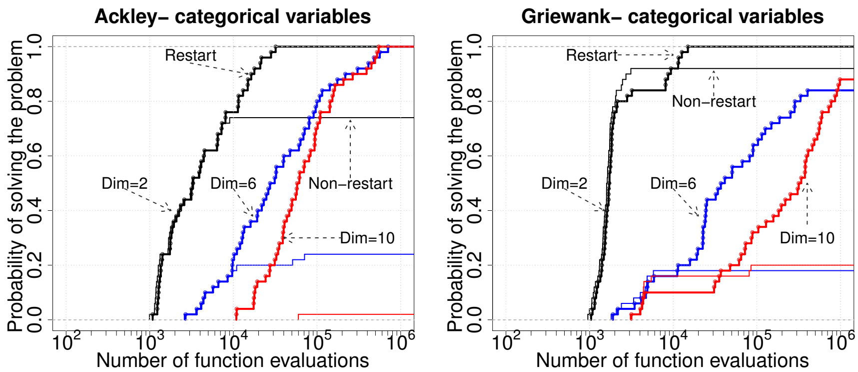

Figure 4.6: The RLDs obtained by ACO$_{\mathbf{MV}}$ with restarts and without restarts. The solution quality demanded is E-10

mark functions with ordered discrete variables, ACO$_{\mathbf{MV}}$ found the optimal solution of $f_{Ackley_{MV}}$, $f_{Rosenbrock_{MV}}$, $f_{Sphere_{MV}}$ and $f_{Griewank_{MV}}$. On the 10 dimensional benchmark functions with categorical variables, ACO$_{\mathbf{MV}}$ found the optimal solution of $f_{Ackley_{MV}}$, $f_{Sphere_{MV}}$ and $f_{Griewank_{MV}}$. With the increase of dimensionality, it is more difficult for ACO$_{\mathbf{MV}}$ to find the optimal solution. Anyway, ACO$_{\mathbf{MV}}$ obtained 100% success rate to solve $f_{Ackley_{MV}}$ and $f_{Sphere_{MV}}$ with both setups on the dimensions (2, 6, 10), and obtained more than 80% success rate to solve $f_{Griewank_{MV}}$ with both setups on the dimensions (2, 6, 10). For a detail level, Figure 4.6 shows a analysis about RLDs of the $f_{Ackley_{MV}}$and $f_{Griewank_{MV}}$ involving categorical variables. The RLD methodology is explained in [47, 69]. Theoretical RLDs can be estimated empirically using multiple independent runs of an algorithm. An empirical RLD provides a graphical view of the development of the probability of finding a solution of a certain quality as a function of time. In the case of stagnation, the probability of finding a solution of a certain quality may be increased by a periodic restart mechanism. The restart criterion of ACO$_{\mathbf{MV}}$ is the number of iterations of ants updating the archive with a relative solution improvement lower than a certain threshold ε. The number of periodic iterations without significant improvement is $MaxStagIter$. $MaxStagIter = 650, \varepsilon = 10^{-5}$ are tuned then used in restart mechanism of ACO$_{\mathbf{MV}}$. As seen from Figure 4.6, the performance of ACO$_{\mathbf{MV}}$ is improved owing to the restart mechanism on fighting against stagnation. With increase of dimensionality from 2 to 6 and 10, the success rate of ACO$_{\mathbf{MV}}$ for solving $f_{Ackley_{MV}}$ still maintains 100%, while the success rate of ACO$_{\mathbf{MV}}$ without restart drops strongly. As for $f_{Griewank_{MV}}$, the success rates of ACO$_{\mathbf{MV}}$ still maintains more than 80% with the increase of dimensionality from 2 to 6 and 10, while the success rate of ACO$_{\mathbf{MV}}$ without restart drops to about 20%.

Table 4.3: Experimental results of ACO$_{\mathbf{MV}}$ with dimensions D = 2, 6, 10. $F1 - F6$ represent $f_{Ellipsoid_{MV}}$, $f_{Ackley_{MV}}$, $f_{Rastrigin_{MV}}$, $f_{Rosenbrock_{MV}}$, $f_{Sphere_{MV}}$ and $f_{Griewank_{MV}}$,respectively. The discrete variables' intervals t = 100. The results are summarized over 50 independent runs, and the values below 1.00E-10 are approximate to 0.00E-10, which is highlighted in **boldface**.

D	Functions	Two upsets of discrete variables							
		Ordered discrete variables				Categorical variables			
		Avg.	Median	Max.	Min.	Avg.	Median	Max.	Min.
2	$F1$	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	$F2$	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	$F3$	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	$F4$	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	$F5$	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	$F6$	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
6	$F1$	8.47e−03	**0.00e+00**	1.65e−01	**0.00e+00**	1.31e+00	4.13e−01	1.26e+01	**0.00e+00**
	$F2$	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	$F3$	1.91+00	1.78e+00	4.38e+00	**0.00e+00**	2.10e+00	2.29e+00	4.38e+00	**0.00e+00**
	$F4$	7.82e−01	**0.00e+00**	1.04e+01	**0.00e+00**	1.00e+01	6.90e+00	5.95e+01	**0.00e+00**
	$F5$	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	$F6$	2.43e−07	**0.00e+00**	1.22e−05	**0.00e+00**	8.41e−04	**0.00e+00**	1.26e−02	**0.00e+00**
10	$F1$	1.99e+00	1.40e+00	1.10e+01	1.17e−01	1.20e+01	7.32e+00	5.48e+01	5.84e−01
	$F2$	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	$F3$	1.37e+01	1.48e+01	2.46e+01	2.93e+00	1.03e+01	9.65e+00	2.03e+01	3.77e+00
	$F4$	1.23e+01	1.32e+01	3.74e+01	**0.00e+00**	4.37e+01	1.91e+01	1.80e+02	1.03e+01
	$F5$	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	$F6$	2.54e−03	**0.00e+00**	4.67e−02	**0.00e+00**	4.52e−03	**0.00e+00**	4.67e−02	**0.00e+00**

Table 4.4: Summary on the classification of engineering optimization problems.

Groups	The type of decision variables
Group I	Continuous variables†
Group II	Continuous and ordered discrete variables
Group III	Continuous and categorical variables
Group IV	Continuous, ordered discrete and categorical variables

† continuous variables should be a particular class of mixed variables with empty set of discrete variables. ACO$_{\mathbf{MV}}$ is also capable to solve continuous optimization.

4.6 Application in Engineering Optimization Problems

We have classified the engineering optimization problems in the literature into 4 groups according to the types of decision variables (see Table 4.4).

Group I include Welded beam design problem case A [17–19, 44, 45, 50, 59, 98]; Group II include pressure vessel design problem [15, 17, 19, 22, 36, 40, 44, 45, 50, 54, 58, 63, 80, 81, 91, 95, 96, 98] and the coil spring design prob-

Table 4.5: Comparison of the best solutions for welded bean design problem case A. The infeasible solutions are highlighted in italics

Methods	$x_1(h)$	$x_2(l)$	$x_3(t)$	$x_4(b)$	$f(x)$
GA1 [17]	0.208800	3.420500	8.997500	0.210000	1.748309
GA2 [19]	0.205986	3.471328	9.020224	0.206480	1.728226
EP [18]	0.205700	3.470500	9.036600	0.205700	1.724852
$(\mu+\lambda)$ES [66]	0.205730	3.470489	9.036624	0.205729	1.724852
CPSO [45]	0.202369	3.544214	9.048210	0.205723	1.728024
HPSO [44]	0.205730	3.470489	9.033624	0.205730	1.724852
NM-PSO [98]	*0.205830*	*3.468338*	*9.033624*	*0.205730*	*1.724717*
PSOLVER [50]	*0.205830*	*3.468338*	*9.033624*	*0.205730*	*1.724717*
SS [59]	0.205729	3.470489	9.033624	0.205730	1.724852
ABC [5]	0.205730	3.470489	9.033624	0.205730	1.724852
$\text{ACO}_{\mathbf{MV}}$	0.205729	3.470489	9.033624	0.205730	1.724852

lem [20, 22, 40, 56, 80, 96]. Group III include the thermal insulation systems design [3, 6, 52]. Group IV include welded beam design problem case B [21, 23, 95]. In this section, we compare the results obtained with those reported in the literature in order to illustrate the performance of $\text{ACO}_{\mathbf{MV}}$. In experimental setup, the tuned parameters configuration on benchmark functions are used. For outstanding the performance of $\text{ACO}_{\mathbf{MV}}$ heuristics and simplifying the algorithm, the most fundamental constraints handling technique, "death penalty", is used. 100 independent runs were performed for each engineering problem. The mathematical formulation of problems are described in Appendix 6.1.

4.6.1 Group I : Welded Beam Design Problem Case A

Recently, many methods previously have been applied into Welded beam design problem case A in Appendix 6.1.1. The best solutions are compared and list in Table 4.5. It should be noted that the results produced by NM-PSO [98] and PSOLVER [50] are infeasible solution because the third constraints had been violated. Table 4.5 illustrates $\text{ACO}_{\mathbf{MV}}$ obtained the best-so-far solution. Table 4.6 illustrates the standard deviation of $\text{ACO}_{\mathbf{MV}}$ results is the smallest and $\text{ACO}_{\mathbf{MV}}$ require the smallest number of functions evaluation, 2303. The successful rate of $\text{ACO}_{\mathbf{MV}}$ for best-so-far solution is 100%. Accordingly, $\text{ACO}_{\mathbf{MV}}$ is the most efficient and robust among the literature in this problem. Additionally, the mean and minimum number of evaluations of $\text{ACO}_{\mathbf{MV}}$ are 2122 and 1888, respectively.

Table 4.6: Statistical results for welded bean design problem case A. The infeasible solutions are highlighted in italics

Methods	f_{Best}	f_{Mean}	f_{worst}	Sd	FEs
GA1 [17]	1.748309	1.771973	1.785835	1.12E-02	N/A
GA2 [19]	1.728226	1.792654	1.993408	7.47E-02	80000
EP [18]	1.724852	1.971809	3.179709	4.43E-01	N/A
$(\mu + \lambda)$ES [66]	1.724852	1.777692	NA	8.80E-02	30000
CPSO [45]	1.728024	1.748831	1.782143	1.29E-02	200000
HPSO [44]	1.724852	1.749040	1.814295	4.01E-02	81000
NM-PSO [98]	*1.724717*	*1.726373*	*1.733393*	*3.50E-03*	*80000*
PSOLVER [50]	*1.724717*	*1.724717*	*1.724717*	*1.62E-11*	*297*
SS [59]	1.724852	1.747429	1.928811	4.67E-02	83703
ABC [5]	1.724852	1.741913	NA	3.10E-02	30000
$\text{ACO}_{\mathbf{MV}}$	1.724852	1.724852	1.724852	1.74E-12	2303

4.6.2 Group II: Pressure Vessel Design Problem Case A, B, C and D

There are four distinctive cases (A, B, C and D) of pressure vessel design problem defined in the literature. These cases differ by the constraints posed on the thickness of the steel used for the heads and the main cylinder. In case A, B, C (see Table 4.7), $\text{ACO}_{\mathbf{MV}}$ obtained the best results in a 100% success rate. The number of evaluations are also the smallest. Case D is more difficult to solve because of the larger range of side constraints for decision variables. It should be noted that the solution of NM-PSO is not feasible for this problem because the values of x_1 and x_2 given for NM-PSO are not integer multiples of 0.0625. Table 4.8 illustrates $\text{ACO}_{\mathbf{MV}}$ obtained the best-so-far solution except the infeasible solution reported by NM-PSO . Table 4.9 illustrates $\text{ACO}_{\mathbf{MV}}$ has 100% success rate to obtain the best-so-far results with smallest standard deviation, which is competitive to PSOLVER. $\text{ACO}_{\mathbf{MV}}$ require 30717 function evaluations. The mean and minimum number of evaluations is 9448 and 1726. PSOLVER is more efficient in the aspect of the number of functions evaluations. However, it should be noted that In [50] PSOLVER is only designed for continuous optimization rather than mixed-variable optimization, therefore, PSOLVER is difficult to solve categorical variables. Moreover, PSOLVER ever reported an infeasible solution in the previous welded beam design problem case A.

4.6.3 Group II: Coil Spring Design Problem

In coil spring design problem, most of the research reported in the literature focused on finding the best solution. Only the recent work by [54] and [20]

Table 4.7: Results for Case A,B,C of the pressure vessel design problem. The mean number of evaluations of the successful runs is given in parentheses.

Case A	[80]	[36]	[54]	ACO$_{\mathbf{MV}}$		
f_{best}	7867.0	7790.588	7019.031	7019.031		
Success rate	100%	99%	89.2%	100%		
FE_s	-	-	10000	1737		
				(1500)		
Case B	[80]	[63]	[96]	[54]	[40]	ACO$_{\mathbf{MV}}$
f_{best}	7982.5	7197.734	7207.497	7197.729	7197.9	7197.729
success rate	100%	90.2%	90.3%	90.2%	-	100%
FE_s	-	-	-	10000	-	1764
						(1470.48)
Case C	[58]	[15]	[91]	[54]	[81]	ACO$_{\mathbf{MV}}$
f_{best}	7127.3	7108.616	7006.9	7006.358	7006.51	7006.358
success rate	100%	99.7%	98.3%	98.3%	-	100%
FE_s	-	-	4800	10000	10000	1666
						(1433.42)

Table 4.8: Comparison of the best solutions for pressure vessel design problem case D. The infeasible solutions are highlighted in italics

Methods	$x_1(T_s)$	$x_2(T_h)$	$x_3(R)$	$x_4(L)$	$f(x)$
GA1 [17]	0.8125	0.4375	40.3239	200.0000	6288.7445
GA2 [19]	0.8125	0.4375	42.0974	176.6540	6059.9463
$(\mu + \lambda)$ES [66]	0.8125	0.4375	42.0984	176.6366	6059.7143
CPSO [45]	0.8125	0.4375	42.0913	176.7465	6061.0777
HPSO [44]	0.8125	0.4375	42.0984	176.6366	6059.7143
RSPSO [95]	0.8125	0.4375	42.0984	176.6366	6059.7143
NM-PSO [98]	*0.8036*	*0.3972*	*41.6392*	*182.4120*	*5930.3137*
PSOLVER [50]	0.8125	0.4375	42.0984	176.6366	6059.7143
ABC [5]	0.8125	0.4375	42.0984	176.6366	6059.7143
ACO$_{\mathbf{MV}}$	0.8125	0.4375	42.0984	176.6366	6059.7143

gave some attention to the number of functions evaluations to reach the best solution. A comparison of the results obtained is presented in Table 4.10. Only [54] and ACO$_{\mathbf{MV}}$ obtained the best-so-far results, 2.65856. The result of [20] is very close to the best-so-far. ACO$_{\mathbf{MV}}$ has the 100% success rate while [54] has a 95% success rate. Though ACO$_{\mathbf{MV}}$ require relative more function evaluations than [54], it is noted that [54] does not consider to handle categorical variables . The mean and minimum of function evaluations

Table 4.9: Statistical results for pressure vessel design problem case D . The mean number of evaluations of the successful runs is given in parentheses. The infeasible solutions are highlighted in italics

Methods	f_{Best}	f_{Mean}	f_{worst}	Sd	FEs
GA1 [17]	6288.7445	6293.8432	6308.1497	7.413E+00	N/A
GA2 [19]	6059.9463	6177.2533	6469.3220	1.309E+02	80000
$(\mu+\lambda)$ES [66]	6059.7143	6379.938037	NA	2.10E+02	30000
CPSO [45]	6061.0777	6147.1332	6363.8041	8.645E+01	200000
HPSO [44]	6059.7143	6099.9323	6288.6770	8.620E+01	81000
RSPSO [95]	6059.7143	6066.2032	6100.3196	1.33E+01	30000
NM-PSO [98]	*5930.3137*	*5946.7901*	*5960.0557*	*9.161E+00*	80000
PSOLVER [50]	6059.7143	6059.7143	6059.7143	4.625E-12	310
ABC [5]	6059.7143	6245.3081	NA	2.05E+02	30000
$\text{ACO}_{\mathbf{MV}}$	6059.7143	6059.7143	6059.7143	3.45E-12	30717 (9448.08)

Table 4.10: Results for the coil spring design problem. The mean number of evaluations of the successful runs is given in parentheses.

	[80]	[16]	[96]	[54]	[40]	[20]	$\text{ACO}_{\mathbf{MV}}$
N	10	9	9	9	9	9	9
D [inch]	1.180701	1.2287	1.227411	1.223041	1.223	1.223044	1.223041
d [inch]	0.283	0.283	0.283	0.283	0.283	0.283	0.283
f_{best}	2.7995	2.6709	2.6681	2.65856	2.659	2.658565	2.65856
success rate	100%	95.4%	95.3%	95.0%	-	<100%	100%
FE_s	-	-	-	8000	-	3711560 (2270994)	19588 (4808.19)

of $\text{ACO}_{\mathbf{MV}}$ are 9948 and 1726. [20] does not report a success rate, but the corresponding objective value vary in the range of $(2.658565, 2.658790)$, the numbers of function evaluation vary in the range of $[539960, 3711560]$.

4.6.4 Group III: Thermal Insulation Systems Design Problem

The thermal insulation systems design problem is one of the few benchmark engineering problems used in the literature that deals with categorical variables. In previous studies, the categorical variables describing the type of insulators used indifferent layers were not considered as optimization variable, but rather as parameters. Only the more recent work of Kokkolaras et al [52] and Abramson et al [3], which are able to handle such categorical variables properly. we show that $\text{ACO}_{\mathbf{MV}}$ can performs comparably to MVP [52] and FMGPS [3]. Table 4.11 present the new best-so-far solution of $\text{ACO}_{\mathbf{MV}}$ after 10000 function evalutations.

4.6.5 Group IV: Welded Beam Design Problem Case B

Welded beam design problem case B is taken from Deb and Goyal [21] and Dimopoulos [23]. It is a variation of case A and is extended to include ordered discrete and categorical variables together. Table 4.12 shows $\text{ACO}_{\mathbf{MV}}$ obtained a new best-so-far solution with a 100% success rate. The maximum, mean and minimum number of evaluations of is 4883, 1436 and 692, respectively. Table 4.14 verifies the best results obtained by $\text{ACO}_{\mathbf{MV}}$ not to violate the constraints.

4.6.6 Related Work on Engineering Optimization Problems

For a detail level analysis on engineering optimization problems, we investigate $\text{ACO}_{\mathbf{MV}}$ RLDs on fighting against stagnation by restart mechanism. An experiment is also conducted to compare the performance of the generic restart mechanism of $\text{ACO}_{\mathbf{MV}}$ with a problem tailored restart mechanism, called *cut-off* restart. The later is based on an approximation of exponential distribution. It is possible to estimate, from an empirically estimated RLD, the number of function evaluations needed to find the required solution with a probability greater than or equal to z if an optimal restart policy is supposed to be used. This estimation is sometimes called computational effort [69, 73] and it is defined as

$$effort = \min(l) \left\{ l \cdot \frac{ln(1-z)}{ln(1-RL_q(l))} \right\} \tag{4.5}$$

The solution l of the computation effort is the *cut-off* evaluations to periodically restart in a simulation of estimate model. $RL_q(l)$ is the algorithm's RLD, defined as $RL_q(l) = P(L_q \leq l)$, where L_q is the random variable representing the number of function evaluations needed to find a solution of quality q, and $P(L_q \leq l)$ is the probability that L_q takes a value less than or equal to l function evaluations. The *cut-off* restart improves the performance of algorithms as seen from Figure 4.7. However we also see that the tuned restart mechanism of $\text{ACO}_{\mathbf{MV}}$ needs less functions evaluations to have 100% success rate than the *cut-off* restart, even if latter one is problem-tailored. Taking the pressure vessel design problem case D of Figure 4.7 for example, the *cut-off* restart starts at the point (2243 function evaluations with a 23% success rate), and obtain a 99% success rate with 44400 function evaluations, while the tuned restart mechanism of $\text{ACO}_{\mathbf{MV}}$

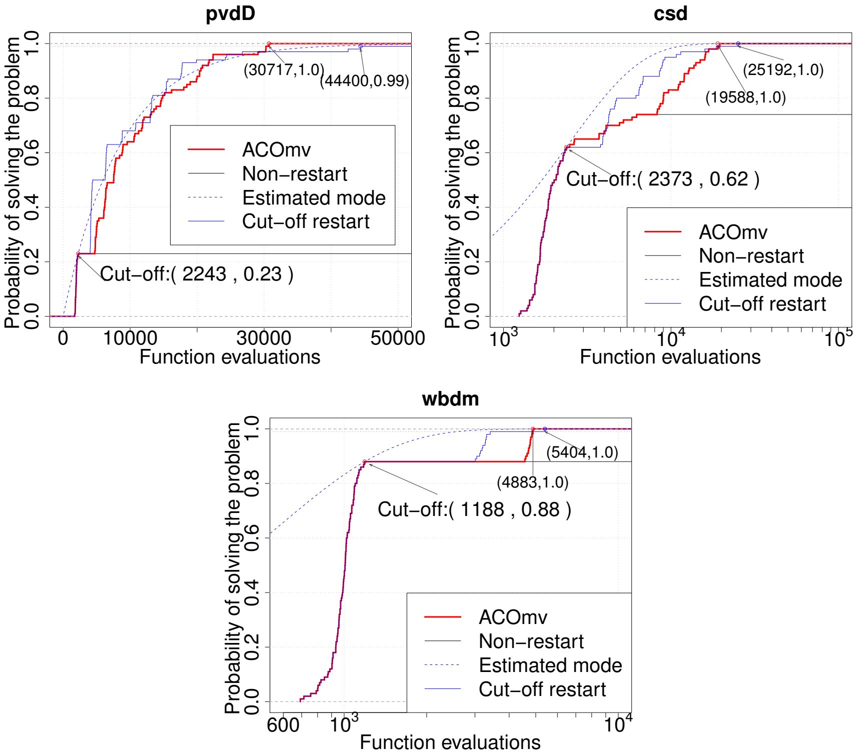

Figure 4.7: the RLDs analysis of $\text{ACO}_{\mathbf{MV}}$ on engineering optimization problems. pvdD is the pressure vessel design problem case D. Csd is the coil spring design problem. WbdB is the welded beam design problem case B

needs 30717 function evaluations to obtain a 100% success rate. Additionally, it is mentioned that in the welded beam design case A and the pressure vessel design problem case A, B and C, we found that $\text{ACO}_{\mathbf{MV}}$ without restarts mechanism has not met any stagnation cases and has 100% success rate to give the best-so-far solution. So, we analyze on the problems, in which the restart mechanism takes effect.

4.7 Conclusions

In this chapter, we have shown how $\text{ACO}_{\mathbb{R}}$ is extended to $\text{ACO}_{\mathbf{MV}}$ for tackling mixed-variable optimization problems. Based on the solution archive framework of $\text{ACO}_{\mathbf{MV}}$, $\text{ACO}_{\mathbf{MV}}$ integrates a component of a continuous optimization solver ($\text{ACO}_{\mathbb{R}}$), a continuous relaxation approach ($\text{ACO}_{\mathbf{MV}}$-o) and a native mixed-variable optimization approach ($\text{ACO}_{\mathbf{MV}}$-c) to solve continuous and mixed-variable optimization problems. In addition, we proposed artificial mixed-variable benchmark functions as well as constructive methods. They provide a sufficiently controlled environment for the investigation of the performance of mixed-variable optimization algorithms, and they provide a training environment for parameter tuning. Based on the benchmark functions, a rigorous comparison between $\text{ACO}_{\textbf{MV}}$-o and $\text{ACO}_{\textbf{MV}}$-c was conducted, so that we can conclude from these results of this comparision that $\text{ACO}_{\textbf{MV}}$-o is better than $\text{ACO}_{\textbf{MV}}$-c in the case that the discrete variables of mixed-variable problems are ordered, while $\text{ACO}_{\textbf{MV}}$-c is better than $\text{ACO}_{\textbf{MV}}$-o in the case that discrete variables of mixed-variable problems are categorical variables. The experiments illustrate the advantage of combining of $\text{ACO}_{\textbf{MV}}$-o and $\text{ACO}_{\textbf{MV}}$-c, and also suggest that discarding of scratching one of them to handle mixed-variable optimization problems is not a good idea. The experimental results for real-world engineering problems illustrate that $\text{ACO}_{\textbf{MV}}$ not only can tackle various classes of decision variables robustly, but also it is efficient in finding high-quality solutions. In the welded beam design case A and the pressure vessel design problem case A, B, C, $\text{ACO}_{\textbf{MV}}$ is the only available algorithm that obtains the best-so-far solution with a 100% success rate as well as the required smallest number of function evaluations. In the pressure vessel design problem case D, $\text{ACO}_{\textbf{MV}}$ obtains the best-so-far solution with a 100% success rate. In the coil spring design problem, $\text{ACO}_{\textbf{MV}}$ is the only one that obtains the best-so-far solution with a 100% success rate. In the thermal insulation systems design problem, $\text{ACO}_{\textbf{MV}}$ obtains the new best-so-far solution. In the welded beam design problem case B, $\text{ACO}_{\textbf{MV}}$ obtained the new best-so-far solution with a 100% success rate and the smallest number of function evaluations.

Table 4.11: Comparison of the best solutions for the thermal insulation systems

Solution information	MVP [52]	FMGPS [3]	$ACO_{\mathbf{MV}}$
Continuous variable			
$x_i(cm)$			
1	0.3125	4.5313	4.9506
2	5.4688	6.7188	7.9729
3	3.9062	4.8437	12.8448
4	6.5625	4.2188	17.07978
5	5.7812	7.3438	9.4420
6	5.1562	9.8438	10.1077
7	13.2812	24.948	0.02811
8	21.4062	12.135	7.3080
9	8.5938	7.5	11.9592
10	9.2188	6.4063	12.1872
11	20.3125	11.5105	6.1197
$T_i(K)$			
1	4.2188	6.125	6.1003
2	7.3438	10.55	11.0841
3	10	14.35	21.2509
4	15	17.994	38.2608
5	20	24.969	51.8508
6	25	36.006	70.1000
7	40	71.094	71.0001
8	71.0938	116.88	99.4475
9	101.25	156.88	153.1701
10	146.25	198.44	236.8358
11	300	300	300
Categorical variable			
I_i			
1	N	N	N
2	N	N	N
3	N	N	N
4	N	N	N
5	N	N	T
6	N	N	E
7	N	T	T
8	E	E	E
9	E	E	E
10	E	T	T
11	T	T	T
Power($\frac{PL}{A}(\frac{W}{cm})$)	25.294	25.58	24.299

Table 4.12: Comparison of the best solutions for welded beam design design problem case B

Methods	$x_1(h)$	$x_2(l)$	$x_3(t)$	$x_4(b)$	$x_5(M)$	$x_6(Joint)$	$f(x)$
GeneAS [21]	0.1875	1.6849	8.2500	0.2500	Steel	4-sided	1.9422
PSOA [23]	0.2500	2.2219	8.2500	0.2500	Steel	2-sided	1.7631
RSPSO [95]	0.1875	1.6842	8.25	0.25	Steel	4-sided	1.9421
$\text{ACO}_{\mathbf{MV}}$	0.225	1.272373	8.25	0.225	Steel	4-sided	1.502942

Table 4.13: Statistical results for welded beam design design problem case B

Methods	f_{Mean}	Sd	FEs
GeneAS [21]	N/A	N/A	N/A
RSPSO [95]	N/A	N/A	N/A
PSOA [23]	1.7631	0	6570
$\text{ACO}_{\mathbf{MV}}$	1.502942	0	1436

Table 4.14: Constrains analysis for welded beam design design problem case B

Constraints	GeneAS [21]	PSOA [23]	RSPSO [95]	$\text{ACO}_{\mathbf{MV}}$
g_1	-0.1621	0	N/A	0
g_2	-380.1660	N/A	-380.1653	-148.8186
g_3	-0.0625	0	N/A	0
g_4	-3.4399	-3.3838	N/A	-3.562618
g_5	-0.0625	-0.1250	N/A	-0.1
g_6	-0.2346	-0.2344	N/A	-0.2349907
g_7	-402.0473	-412.5254	N/A	-1630.64

Chapter 5

Conclusions and Future Work

5.1 Conclusions

In this thesis, we have proposed two improved ant colony optimization algorithms for continuous and mixed discrete-continuous optimization problems. These are IACO$_{\mathbb{R}}$-LS and ACO$_{\mathbf{MV}}$, respectively.

In Chapter 2, based on the new C++ implementation of ACO$_{\mathbb{R}}$ and Sep-ACO$_{\mathbb{R}}$, we further investigated their performance and addressed their possible drawbacks.

Then, we proposed IACO$_{\mathbb{R}}$-LS, an ACO$_{\mathbb{R}}$ algorithm with growing solution archive hybridized with a local search procedure in Chapter 3. Three different local search procedures, Powell's conjugate directions set, Powell's BOBYQA, and Mtsls1, were tested with IACO$_{\mathbb{R}}$-LS in order to enhance its search intensification. The very good performance of IACO$_{\mathbb{R}}$-Mtsls1 is a clear indication of the high potential hybrid ACO algorithms have for the continuous domain. In fact, IACO$_{\mathbb{R}}$-Mtsls1 is clearly competitive with state-of-the-art continuous optimizers.

In Chapter 4 , we have shown how ACO$_{\mathbb{R}}$ is extended to ACO$_{\mathbf{MV}}$ for tackling mixed-variable optimization problems. Based on the solution archive framework of ACO$_{\mathbf{MV}}$, ACO$_{\mathbf{MV}}$ integrates a component of a continuous optimization solver (ACO$_{\mathbb{R}}$), a continuous relaxation approach (ACO$_{\mathbf{MV}}$-o) and a native mixed-variable optimization approach (ACO$_{\mathbf{MV}}$-c) to solve continuous and mixed-variable optimization problems. In addition, we proposed artificial mixed-variable benchmark functions as well as constructive methods. They provide a sufficiently controlled environment for the investigation of the performance of mixed-variable optimization algorithms, and they provide a training environment for parameter tuning. Based on the benchmark functions, a rigorous comparison between ACO$_{\mathbf{MV}}$-o and ACO$_{\mathbf{MV}}$-c was conducted, so that we can conclude from these results of this comparision that $\text{ACO}_{\textbf{MV}}$-o is better than $\text{ACO}_{\textbf{MV}}$-c in the case that the discrete variables of mixed-variable problems are ordered, while $\text{ACO}_{\textbf{MV}}$-c is better than $\text{ACO}_{\textbf{MV}}$-o in the case that discrete variables of mixed-variable problems are categorical variables. The experiments illustrate the advantage of combining of $\text{ACO}_{\textbf{MV}}$-o and $\text{ACO}_{\textbf{MV}}$-c, and also suggest that discarding of scratching one of them to handle mixed-variable optimization problems is not a good idea. The experimental results for real-world engineering problems illustrate that $\text{ACO}_{\textbf{MV}}$ not only can tackle various classes of decision variables robustly, but also it is efficient in finding high-quality solutions. In the welded beam design case A and the pressure vessel design problem case A, B, C, $\text{ACO}_{\textbf{MV}}$ is the only available algorithm that obtains the best-so-far solution with a 100% success rate as well as the required smallest number of function evaluations. In the pressure vessel design problem case D, $\text{ACO}_{\textbf{MV}}$ obtains the best-so-far solution with a 100% success rate. In the coil spring design problem, $\text{ACO}_{\textbf{MV}}$ is the only one that obtains the best-so-far solution with a 100% success rate. In the thermal insulation systems design problem, $\text{ACO}_{\textbf{MV}}$ obtains the new best-so-far solution. In the welded beam design problem case B, $\text{ACO}_{\textbf{MV}}$ obtained the new best-so-far solution with a 100% success rate and the smallest number of function evaluations.

5.2 Future Work

In practice, high dimensional and highly variable-correlated continuous optimization problems also need to be optimized. Therefore, a new effective and efficient variable correlation method for $\text{ACO}_{\mathbb{R}}$ is one of our ongoing works. Moreover, for implementing a high-performing continuous algorithm, we are investigating and fairly benchmarking state-of-the-art continuous optimization algorithms. Since, different continuous optimization algorithms may be preferably depending on the characteristics of problems, one promising direction we intend to research on is to automatically select and configure continuous optimizers from components, thereby facing challenging continuous instances with different characteristics.

In the aspect of ACO for mixed discrete-continuous optimization, the solution archive of $\text{ACO}_{\textbf{MV}}$ consists in a flexible framework that allows to bring in a resizing population strategy and a hybrid with a subsidiary local search procedure. The incremental population social learning mechanism with local search [60, 68, 70, 71] in Chaper 3 is an interesting modification for $\text{ACO}_{\textbf{MV}}$. Powell's conjugate directions set [76], Powell's BOBYQA [77], and Lin-Yu Tseng's Mtsls1 methods [93] and Hansen's CMA-ES [41] are being considered to be hybridized with $\text{ACO}_{\textbf{MV}}$ for continuous variables and ordered discrete variables. Some typical local search in discrete optimization are considered for handling categorical variables. We also intend to develop an effective constraint-handling technique based on the $\mathrm{ACO}_{\mathbf{MV}}$ framework to tackle highly challenging constrained mixed-variable optimization applications. Finally, a tuning-in-the-loop approach [71] is to be used to redesign $\mathrm{ACO}_{\mathbf{MV}}$. A promising application of $\mathrm{ACO}_{\mathbf{MV}}$ is that the heuristics of $\mathrm{ACO}_{\mathbf{MV}}$ meet the variables arising in the algorithm configuration problem [13], in which typically not only the setting of numerical parameters but also that of categorical parameters needs to be determined. Recently, in [97], several continuous algorithms have been used with F-race [12] to automatically tune parameters from real variable and large ordinal integer variables. $\mathrm{ACO}_{\mathbf{MV}}$ with F-race to tackle mixed-variable parameters including categorical variable is also our following work.

Chapter 6

Appendix

6.1 Mathematical formulation of engineering problems

6.1.1 Welded Beam Design Problem Case A

The mathematical formulation of the welded beam design problem is given in Table 6.1. The schema is shown in Figure 6.1

6.1.2 Welded Beam Design Problem Case B

The welded beam design problem case B is a variation of case A. It is extended to include two types of welded joint configuration and four possible beam materials. The changed places with respect to the formulation in Table 6.1 are shown in Equation 6.1.

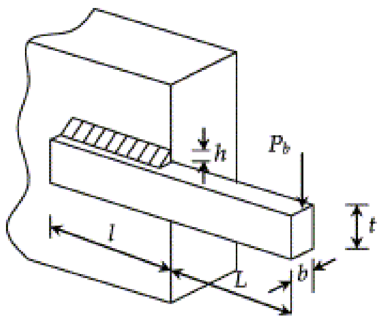

Figure 6.1: Schematic of welded beam design problem case A.

Table 6.1: The mathematical formulation of the welded beam design problem case A.

	$\min f(\vec{x}) = 1.10471\, x_1^2 x_2 + 0.04811\, x_3 x_4\, (14 + x_2)$
g_1	$\tau(\vec{x}) - \tau_{max} \leq 0$
g_2	$\sigma(\vec{x}) - \sigma_{max} \leq 0$
g_3	$x_1 - x_4 \leq 0$
g_4	$0.10471\, x_1^2 + 0.04811\, x_3 x_4\, (14 + x_2) - 5 \leq 0$
g_5	$0.125 - x_1 \leq 0$
g_6	$\delta(\vec{x}) - \delta_{max} \leq 0$
g_7	$P - P_c(\vec{x}) \leq 0$
g_8	$0.1 \leq x_1, x_4 \leq 2.0$
g_9	$0.1 \leq x_2, x_3 \leq 10.0$
where	$\tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$ $\tau' = \frac{P}{\sqrt{2}x_1 x_2}, \tau'' = \frac{MR}{J}, M = P(L + \frac{X_2}{2})$ $R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1+x_3}{2})^2}$ $J = 2\left\{\sqrt{2}x_1 x_2 \left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}$ $\sigma(\vec{x}) = \frac{6PL}{x_4 x_3^2}, \delta(x) = \frac{4PL^3}{E x_3^3 x_4}$ $P_c(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2 x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$ $P = 6000lb, L = 14in., E = 30 \times 10^6 psi, G = 12 \times 10^6 psi$ $\tau_{max} = 1360psi, \sigma_{max} = 30000psi, \delta = 0.25in.$

Table 6.2: Material properties for the welded beam design problem case B

Methods x_5	$S(10^3 psi)$	$E(10^6 psi)$	$G(10^6 psi)$	c_1	c_2
Steel	30	30	12	0.1047	0.0481
Cast iron	8	14	6	0.0489	0.0224
Aluminum	5	10	4	0.5235	0.2405
Brass	8	16	6	0.5584	0.2566

$$
\begin{aligned}
&\min f(\vec{x}) = (1 + c_1)\, x_1^2 x_2 + c_2\, x_3 x_4\, (14 + x_2) \\
&S - \tau_{max} \leq 0 \\
&J = 2\left\{\sqrt{2}x_1 x_2 \left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}, \text{ if } x_6 : \text{two side} \\
&J = 2\left\{\sqrt{2}x_1 x_2 \left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]\right\}, \text{ if } x_6 : \text{four side} \\
&\tau_{max} = 0.577 \cdot S
\end{aligned}
\tag{6.1}
$$

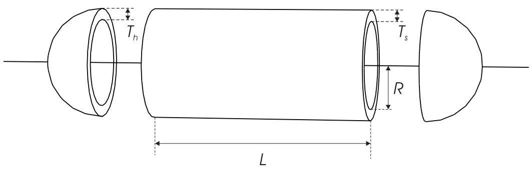

Figure 6.2: Schema of the pressure vessel to be designed.

Table 6.3: The mathematical formulation of the cases (A, B, C and D) of the pressure vessel design problem.

No	Case A	Case B	Case C	Case D
	$\min f = 0.6224\,T_s RL + 1.7781\,T_h R^2 + 3.1611\,T_s^2 L + 19.84\,T_s^2 R$			
g_1	$-T_s + 0.0193\,R \leq 0$			
g_2	$-T_h + 0.00954\,R \leq 0$			
g_3	$-\pi\,R^2 L - \frac{4}{3}\pi\,R^3 + 750 \cdot 1728 \leq 0$			
g_4	$L - 240 \leq 0$			
g_5	$1.1 \leq T_s \leq 12.5$	$1.125 \leq T_s \leq 12.5$	$1 \leq T_s \leq 12.5$	$0 \leq T_s \leq 100$
g_6	$0.6 \leq T_h \leq 12.5$	$0.625 \leq T_h \leq 12.5$		$0 \leq T_h \leq 100$
g_7	$0.0 \leq R \leq 240$			$10 \leq R \leq 200$
g_8	$0.0 \leq L \leq 240$			$10 \leq L \leq 200$

6.1.3 Pressure Vessel Design Problems

The pressure vessel design problems requires designing a pressure vessel consisting of a cylindrical body and two hemispherical heads such that the cost of its manufacturing is minimized subject to certain constraints. The schematic picture of the vessel is presented in Figure 6.2. There are four variables where values must be chosen: the thickness of the main cylinder T_s, the thickness of the heads T_h, the inner radius of the main cylinder R, and the length of the main cylinder L. While variables R and L are continuous, the thickness for variables T_s and T_h may be chosen only from a set of allowed values, these being the integer multiples of 0.0625 inch. The mathematical formulation for the cases (A, B, C and D) is given in Table 6.3.

6.1.4 Coil Spring Design Problem

The problem consists in designing a helical compression spring that will hold an axial and constant load. The objective is to minimize the volume of the spring wire used to manufacture the spring. A schematic of the coil spring to be designed is shown in Figure 6.3. The decision variables are the number of spring coils N, the outside diameter of the spring D, and the spring wire diameter d. The number of coils N is an integer variable, the outside diameter of the spring D is a continuous variable, and finally, the spring wire diameter is a discrete variable, whose possible values are given

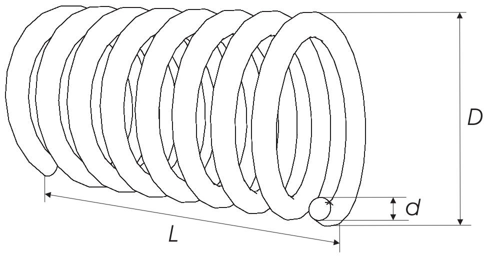

Figure 6.3: Schematic of the coil spring to be designed.

Table 6.4: Standard wire diameters available for the spring coil.

Allowed wire diameters [inch]					
0.0090	0.0095	0.0104	0.0118	0.0128	0.0132
0.0140	0.0150	0.0162	0.0173	0.0180	0.0200
0.0230	0.0250	0.0280	0.0320	0.0350	0.0410
0.0470	0.0540	0.0630	0.0720	0.0800	0.0920
0.1050	0.1200	0.1350	0.1480	0.1620	0.1770
0.1920	0.2070	0.2250	0.2440	0.2630	0.2830
0.3070	0.3310	0.3620	0.3940	0.4375	0.5000

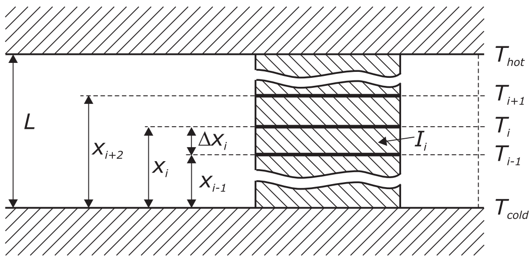

Figure 6.4: Schematic of the thermal insulation system.

in Table 6.4. The mathematical formulation is in Table 6.5. The penalty function was defined in Equation 6.2, which is the similar to [56] for a more rigorous heuristics comparison between $\mathrm{ACO}_{\mathbf{MV}}$ and Differential Evolution.

$$
\begin{aligned}
&f = f_c \prod_{i=1}^{8} c_i^3, \\
&c_i = \begin{cases} 1 + s_i g_i & \text{if } g_i > 0, \\ 1 & \text{otherwise,} \end{cases} \\
&s_1 = 10^{-5},\ s_2 = s_4 = s_6 = 1,\ s_3 = s_5 = s_7 = s_8 = 10^2.
\end{aligned}
\tag{6.2}
$$

Table 6.5: The mathematical formulation for the coil spring design problem.

	$\min f_c(N, D, d) = \frac{\pi^2 Dd^2(N+2)}{4}$
No	Constraint
g_1	$\frac{8\,C_f F_{\max} D}{\pi\, d} - S \leq 0$
g_2	$l_f - l_{\max} \leq 0$
g_3	$d_{\min} - d \leq 0$
g_4	$D - D_{\max} \leq 0$
g_5	$3.0 - \frac{D}{d} \leq 0$
g_6	$\sigma_p - \sigma_{pm} \leq 0$
g_7	$\sigma_p + \frac{F_{\max}-F_p}{K} + 1.05(N+2)d - l_f \leq 0$
g_8	$\sigma_w - \frac{F_{\max}-F_p}{K} \leq 0$
where	$C_f = \frac{4\frac{D}{d}-1}{4\frac{D}{d}-4} + \frac{0.615\,d}{D}$ $K = \frac{Gd^4}{8\,ND^3}$ $\sigma_p = \frac{F_p}{K}$ $l_f = \frac{F_{\max}}{K} + 1.05(N+2)d$

6.1.5 Thermal Insulation Systems Design Problem

The schema is shown in Figure 6.4. The basic mathematical formulation of the classic model of thermal insulation systems is defined in Table 6.6. The effective thermal conductivity k of all these insulators varies with the temperature and does so differently for different materials. Considering that the number of intercepts n is defined in advance, and based on the presented model, we may define the following problem variables:

- $I_i \in \mathbf{M},\ i = 1, ..., n+1$ — the material used for the insulation between the $(i-1)$-st and the i-th intercepts (from a set $\mathbf{M}$ of materials).
- $\Delta x_i \in \mathbb{R}_+,\ i = 1, ..., n+1$ — the thickness of the insulation between the $(i-1)$-st and the i-th intercepts.
- $\Delta T_i \in \mathbb{R}_+,\ i = 1, ..., n+1$ — the temperature difference of the insulation between the $(i-1)$-st and the i-th intercepts.

This way, there are $n+1$ categorical variables chosen form a set $\mathbf{M}$ of available materials. The remaining $2n+2$ variables are continuous.

Table 6.6: The mathematical formulation for the coil spring design problem.

$f(\mathbf{x}, \mathbf{T}) = \sum_{i=1}^{n} P_i = AC_i \left(\frac{T_{\text{hot}}}{T_i} - 1\right) \left(\frac{\int_{T_i}^{T_{i+1}} kdT}{\Delta x_i} - \frac{\int_{T_{i-1}}^{T^i} kdT}{\Delta x_{i-1}}\right)$	
No	Constraint
g_1	$\Delta x_i \geq 0,\ i = 1, ..., n+1$
g_2	$T_{\text{cold}} \leq T_1 \leq T_2 \leq ... \leq T_{n-1} \leq T_n \leq T_{\text{hot}}$
g_3	$\sum_{i=1}^{n+1} \Delta x_i = L$
where	$C = 2.5 \quad \text{if} \;\; T \geq 71\,\text{K}$ $C = 4 \quad \text{if} \;\; 71\,\text{K} > T > 4.2\,\text{K}$ $C = 5 \quad \text{if} \;\; T \leq 4.2\,\text{K}$

Bibliography

[1] K. Abhishek, S. Leyffer, and J.T. Linderoth. Modeling without categorical variables: a mixed-integer nonlinear program for the optimization of thermal insulation systems. *Optimization and Engineering*, 11(2):185–212, 2010.

[2] M.A. Abramson. *Pattern search algorithms for mixed variable general constrained optimization problems.* PhD thesis, Rice University, USA, 2002.

[3] M.A. Abramson. Mixed variable optimization of a load-bearing thermal insulation system using a filter pattern search algorithm. *Optimization and Engineering*, 5(2):157–177, 2004.

[4] M.A. Abramson, C. Audet, J.W. Chrissis, and J.G. Walston. Mesh adaptive direct search algorithms for mixed variable optimization. *Optimization Letters*, 3(1):35–47, 2009.

[5] B. Akay and D. Karaboga. Artificial bee colony algorithm for large-scale problems and engineering design optimization. *Journal of Intelligent Manufacturing*, pages 1–14, 2010.

[6] C. Audet and J.E. Dennis Jr. Pattern search algorithms for mixed variable programming. *SIAM Journal on Optimization*, 11(3):573–594, 2001.

[7] A. Auger and N. Hansen. A restart CMA evolution strategy with increasing population size. In *Proceedings of IEEE Congress on Evolutionary Computation, CEC 2005*, pages 1769–1776, Piscataway, NJ, 2005. IEEE Press.

[8] A. Auger, N. Hansen, J. Perez Zerpa, R. Ros, and M. Schoenauer. Experimental comparisons of derivative free optimization algorithms. In Jan Vahrenhold, editor, *Experimental Algorithms*, volume 5526 of *LNCS*, pages 3–15, Berlin, Germany, 2009. Springer.

[9] P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the F-Race algorithm: Sampling design and iterative refinement. In

Proceedings of the 4th international conference on Hybrid metaheuristics, volume 4771 of *LNCS*, pages 108–122, Berlin, Germany, 2007. Springer.

[10] G. Bilchev and I.C. Parmee. The ant colony metaphor for searching continuous design spaces. In Terence C. Fogarty, editor, *Proceedings of the AISB Workshop on Evolutionary Computation*, volume 993 of *LNCS*, pages 25–39. Springer-Verlag, Berlin, Germany, 1995.

[11] M. Birattari. *The Problem of Tuning Metaheuristics as Seen From a Machine Learning Perspective. PhD thesis*, volume 292 of *Dissertationen zur Künstlichen Intelligenz*. Akademische Verlagsgesellschaft Aka GmbH, Berlin, Germany, 2005.

[12] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In W. B. Langdon *et al.*, editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 11–18. Morgan Kaufman, San Francisco, CA, 2002.

[13] M. Birattari, Z. Yuan, P. Balaprakash, and T. Stützle. F-Race and iterated F-Race: An overview. *Experimental Methods for the Analysis of Optimization Algorithms*, pages 311–336, 2010.

[14] Christian Blum, M. Yábar Vallès, and María J. Blesa. An ant colony optimization algorithm for DNA sequencing by hybridization. *Computers & Operations Research*, 35(11):3620–3635, 2008.

[15] Y.J. Cao and Q.H. Wu. Mechanical design optimization by mixed-variable evolutionary programming. In *Proceedings of IEEE Conference on Evolutionary Computation*, pages 443–446, Piscataway, NJ, 1997. IEEE Press.

[16] J.L. Chen and Y.C. Tsao. Optimal design of machine elements using genetic algorithms. *CHUNG-KUO CHI HSUEH KUNG CH'ENG HSUEH PAO.*, 14(2):193–199, 1993.

[17] C. Coello and A. Carlos. Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2):113–127, 2000.

[18] C.A.C. Coello and R.L. Becerra. Efficient evolutionary optimization through the use of a cultural algorithm. *Engineering Optimization*, 36(2):219–236, 2004.

[19] C.A. Coello Coello and E. Mezura Montes. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, 16(3):193–203, 2002.

[20] D. Datta and J. Figueira. A Real-Integer-Discrete-Coded Differential Evolution Algorithm: A Preliminary Study. *Evolutionary Computation in Combinatorial Optimization*, pages 35–46, 2010.

[21] K. Deb and M. Goyal. A combined genetic adaptive search (GeneAS) for engineering design. *Computer Science and Informatics*, 26:30–45, 1996.

[22] K. Deb and M. Goyal. A flexible optimiztion procedure for mechanical component design based on genetic adaptive search. *Journal of Mechanical Design*, 120(2):162–164, 1998.

[23] G.G. Dimopoulos. Mixed-variable engineering optimization based on evolutionary and social metaphors. *Computer methods in applied mechanics and engineering*, 196(4-6):803–817, 2007.

[24] M. Dorigo. *Optimization, Learning and Natural Algorithms* (in Italian). PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

[25] M. Dorigo, V. Maniezzo, and A. Colorni. Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41, 1996.

[26] M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, Cambridge, MA, 2004.

[27] Marco Dorigo and Christian Blum. Ant colony optimization theory: A survey. *Theoretical Computer Science*, 344(2-3):243–278, 2005.

[28] Marco Dorigo, G. Di Caro, and L. M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*, 5(2):137–172, 1999.

[29] Marco Dorigo and L. M. Gambardella. Ant Colony System: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66, 1997.

[30] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41, 1996.

[31] Marco Dorigo and Thomas Stützle. Ant colony optimization: Overview and recent advances. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 227–263. Kluwer Academic Publishers, Boston, MA, 2010.

[32] J. Dréo and P. Siarry. A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Proceedings of the Third International Workshop on Ant Algorithms, ANTS 2002*, volume 2463 of *LNCS*, pages 216–221, Berlin, Germany, 2002. Springer-Verlag.

[33] J. Dréo and P. Siarry. Continuous interacting ant colony algorithm based on dense heterarchy. *Future Generation Computer Systems*, 20(5):841–856, 2004.

[34] A.E. Eiben and T. Bäck. Empirical investigation of multiparent recombination operators in evolution strategies. *Evolutionary Computation*, 5(3):347–365, 1997.

[35] L.J. Eshelman and J.D. Schaffer. Real-coded genetic algorithms and interval-schemata. *Foundations of Genetic Algorithms*, 2(1993):187–202, 1993.

[36] J.-F. Fu, R.G. Fenton, and W.L. Cleghorn. A mixed integer-discrete-continuous programming method and its application to engineering design optimization. *Engineering Optimization*, 17(4):263–280, 1991.

[37] L. M. Gambardella, Éric D. Taillard, and G. Agazzi. MACS-VRPTW: A multiple ant colony system for vehicle routing problems with time windows. In David Corne, Marco Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 63–76. McGraw Hill, London, UK, 1999.

[38] D.E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-wesley, 1989.

[39] G.H. Golub and C.F. Van Loan. *Matrix computations*. Johns Hopkins University Press, Baltimore, MD, 1989.

[40] C. Guo, J. Hu, B. Ye, and Y. Cao. Swarm intelligence for mixed-variable design optimization. *Journal of Zhejiang University Science*, 5(7):851–860, 2004.

[41] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[42] N. Hansen, A. Ostermeier, and A. Gawelczyk. On the adaptation of arbitrary normal mutation distributions in evolution strategies: The generating set adaptation. In *Proc. of 6th ICGA*, pages 57–64, San Francisco, CA, 1995. Morgan Kaufmann.

[43] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, and A. Auger. PSO facing non-separable and ill-conditioned problems. Technical report, INRIA, 2008.

[44] Q. He and L. Wang. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. *Applied Mathematics and Computation*, 186(2):1407–1422, 2007.

[45] Q. He and L. Wang. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence*, 20(1):89–99, 2007.

[46] F. Herrera, M. Lozano, and D. Molina. Test suite for the special issue of soft computing on scalability of evolutionary algorithms and other metaheuristics for large scale continuous optimization problems, 2010. URL: `http://sci2s.ugr.es/eamhco/updated-functions1-19.pdf`.

[47] H.H. Hoos and T. Stützle. *Stochastic local search: Foundations and applications*. Morgan Kaufmann, San Francisco, CA, 2005.

[48] R. Ihaka and R. Gentleman. R: A language for data analysis and graphics. *Journal of computational and graphical statistics*, 5(3):299–314, 1996.

[49] S.G. Johnson. The NLopt nonlinear-optimization package, 2009.

[50] A.H. Kayhan, H. Ceylan, M.T. Ayvaz, and G. Gurarslan. PSOLVER: A new hybrid particle swarm optimization algorithm for solving continuous optimization problems. *Expert Systems with Applications*, 37(10):6798–6808, 2010.

[51] J. Kennedy, R.C. Eberhart, and Y. Shi. *Swarm intelligence*. Morgan Kaufmann, San Francisco, CA, 2001.

[52] M. Kokkolaras, C. Audet, and J.E. Dennis Jr. Mixed variable optimization of the number and composition of heat intercepts in a thermal insulation system. *Optimization and Engineering*, 2(1):5–29, 2001.

[53] O. Korb, T. Stützle, and T.E. Exner. An ant colony optimization approach to flexible protein–ligand docking. *Swarm Intelligence*, 1(2):115–134, 2007.

[54] J. Lampinen and I. Zelinka. Mechanical engineering design optimization by differential evolution. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 127–146. McGraw-Hill, London, UK, 1999.

[55] J. Lampinen and I. Zelinka. Mixed integer-discrete-continuous optimization by differential evolution. Part 1: the optimization method. In P. Ošmera, editor, *Proceedigns of MENDEL'99, 5th International Mendel Conference of Soft Computing*, pages 71–76. Brno University of Technology, Brno, Czech Repuplic, 1999.

[56] J. Lampinen and I. Zelinka. Mixed integer-discrete-continuous optimization by differential evolution. Part 2: a practical example. In P. Ošmera, editor, *Proceedigns of MENDEL'99, 5th International Mendel Conference of Soft Computing*, pages 77–81. Brno University of Technology, Brno, Czech Repuplic, 1999.

[57] G. Leguizamón and C.A.C. Coello. An alternative $ACO_{\mathbb{R}}$ algorithm for continuous optimization problems. In M. Dorigo et al., editors, *Proc of ANTS 2010*, volume 6234 of *LNCS*, pages 48–59, Germany, 2010. Springer.

[58] H.-L. Li and C.-T. Chou. A global approach for nonlinear mixed discrete programing in design optimization. *Engineering Optimization*, 22:109–122, 1994.

[59] H.S. Li and S.K. Au. Solving Constrained Optimization Problems via Subset Simulation. In Rafi L. Muhanna Michael Beer and Robert L. Mullen, editors, *Proceedings of 4th International Workshop on Reliable Engineering Computing*, 2010.

[60] T. Liao, Marco A. Montes de Oca., D. Aydın, T. Stützle, and M. Dorigo. An Incremental Ant Colony Algorithm with Local Search for Continuous Optimization. In *Proceedings of Genetic and Evolutionary Computation Conference, GECCO 2011*, New York, 2011. ACM Press. accepted.

[61] T. Liao, Marco A. Montes de Oca., and T. Stützle. Tuning Parameters across Mixed Dimensional Instances: A Performance Scalability Study of Sep-G-CMA-ES. In *Proceedings of the Workshop on Scaling Behaviours of Landscapes, Parameters and Algorithms of the Genetic and Evolutionary Computation Conference, GECCO 2011*, New York, 2011. ACM Press. accepted.

[62] C. Ling, S. Jie, Q. Ling, and C. Hongjian. A method for solving optimization problems in continuous space using ant colony algorithm. In *Proceedings of International Workshop on Ants Algorithms, ANTS 2002*, volume 2463 of *LNCS*, pages 288–289, Berlin, Germany, 2002. Springer.

[63] H.T. Loh and P.Y. Papalambros. Computation implementation and test of a sequential linearization approach for solving mixed-discrete nonlinear design optimization. *Journal of Mechanical Design*, 113(3):335–345, 1991.

[64] M. Lozano, D. Molina, and F. Herrera. Editorial: Scalability of evolutionary algorithms and other metaheuristics for large-scale continuous optimization problems. *Soft Computing*, 2011. In Press.

[65] S. Lucidi, V. Piccialli, and M. Sciandrone. An algorithm model for mixed variable programming. *SIAM Journal on Optimization*, 15(4):1057–1084, 2005.

[66] E. Mezura-Montes and C. Coello. Useful infeasible solutions in engineering optimization with evolutionary algorithms. In *Mexican International Conference on Artificial Intelligence: Advances in Artificial Intelligence, MICAI 2005*, volume 3789 of *LNCS*, pages 652–662. Springer, 2005.

[67] N. Monmarché, G. Venturini, and M. Slimane. On how Pachycondyla apicalis ants suggest a new search algorithm. *Future Generation Computer Systems*, 16(9):937–946, 2000.

[68] M.A. Montes De Oca, K. Enden, and T. Stützle. Incremental Particle Swarm-Guided Local Search for Continuous Optimization. In *Proceedings of the 5th International Workshop on Hybrid Metaheuristics, HM 2008*, volume 5296 of *LNCS*, pages 72–86, Berlin, Germany, 2008. Springer-Verlag.

[69] MA Montes de Oca, T. Stutzle, M. Birattari, and M. Dorigo. Frankenstein's PSO: a composite particle swarm optimization algorithm. *IEEE Transactions on Evolutionary Computation*, 13(5):1120–1132, 2009.

[70] MA Montes de Oca, T. Stützle, K. Van den Enden, and M. Dorigo. Incremental social learning in particle swarms. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 41(2):368–384, 2011.

[71] Marco A. Montes de Oca, Doğan Aydın, and Thomas Stützle. An incremental particle swarm for large-scale optimization problems: An example of tuning-in-the-loop (re)design of optimization algorithms. *Soft Computing*, 2011. In press.

[72] Marco A. Montes de Oca and Thomas Stützle. Towards incremental social learning in optimization and multiagent systems. In W. Rand et al., editors, *Workshop ECMSS. GECCO 2008*, pages 1939–1944. ACM Press, New York, 2008.

[73] Jens Niehaus and Wolfgang Banzhaf. More on computational effort statistics for genetic programming. In *Proceedings of the 6th European conference on Genetic programming*, EuroGP'03, pages 164–172, Berlin, 2003. Springer-Verlag.

[74] J. Ocenasek and J. Schwarz. Estimation Distribution Algorithm for Mixed Continuous-discrete Optimization Problems. *Intelligent technologies:theory and applications: new trends in intelligent technologies*, page 227, 2002. IOS Press.

[75] S.H. Pourtakdoust and H. Nobahari. An extension of ant colony system to continuous optimization problems. In M. Dorigo et al., editors, *Proceedings of 4th International Workshop on Ant Colony Optimization and Swarm Intelligence, ANTS 2004*, volume 3172 of *LNCS*, pages 158–173, Germany, 2004. Springer.

[76] M.J.D. Powell. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155, 1964.

[77] MJD Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. Technical report, Department of Applied Mathematics and Theoretical Physics, University of Cambridge, 2009.

[78] S. Praharaj and S. Azarm. Two-level non-linear mixed discrete/ continuous optimization-based design: An application to printed circuit board assemblies. *Advances in Design Automation*, 1(44):307–321, 1992.

[79] S.S. Rao and Y. Xiong. A Hybrid Genetic Algorithm for Mixed-Discrete Design Optimization. *Journal of Mechanical Design*, 127:1100, 2005.

[80] E. Sandgren. Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design*, 112:223–229, 1990.

[81] H. Schmidt and G. Thierauf. A combined heuristic optimization technique. *Advances in Engineering Software*, 36:11–19, 2005.

[82] K. Socha and Christian Blum. An ant colony optimization algorithm for continuous optimization: An application to feed-forward neural network training. *Neural Computing & Applications*, 16(3):235–247, 2007.

[83] K. Socha and M. Dorigo. Ant colony optimization for continuous domains. *European Journal of Operational Research*, 185(3):1155–1173, 2008.

[84] M.A. Stelmack and S.M. Batill. Concurrent subspace optimization of mixed continuous/discrete systems. In *Proceedings of AIAA/ASME/ASCE/AHS/ASC 38th Structures, Structural Dynamic and Materials Conference*. AIAA, Reston, VA, 1997.

[85] R. Storn and K. Price. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359, 1997.

[86] Thomas Stützle. An ant approach to the flow shop problem. In *Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98)*, volume 3, pages 1560–1564. Verlag Mainz, Aachen, Germany, 1998.

[87] Thomas Stützle and Holger H. Hoos. $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System. *Future Generation Computer Systems*, 16(8):889–914, 2000.

[88] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, YP Chen, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Technical Report 2005005, Nanyang Technological University, 2005.

[89] R.S. Sutton and A.G. Barto. *Introduction to reinforcement learning.* MIT Press, Cambridge, MA, 1998.

[90] K. Tang, X. Yao, PN Suganthan, C. MacNish, YP Chen, CM Chen, and Z. Yang. Benchmark functions for the CEC 2008 special session and competition on large scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China, 2007. URL: `http://nical.ustc.edu.cn/cec08ss.php`.

[91] G. Thierauf and J. Cai. Evolution strategies—parallelization and application in engineering optimization. In B.H.V. Topping, editor, *Parallel and distributed processing for computational mechanics: systems and tools*, pages 329–349. Saxe-Coburg Publications, Edinburgh, UK, 2000.

[92] V. Torczon et al. On the convergence of pattern search algorithms. *SIAM Journal on optimization*, 7(1):1–25, 1997.

[93] L.Y. Tseng and C. Chen. Multiple trajectory search for large scale global optimization. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2008*, pages 3052–3059, Piscataway, NJ, 2008. IEEE Press.

[94] N. Turkkan. Discrete optimization of structures using a floating point genetic algorithm. In *Proceedings of Annual Conference of the Canadian Society for Civil Engineering*, pages 4–7, 2003.

[95] J. Wang and Z. Yin. A ranking selection-based particle swarm optimizer for engineering design optimization problems. *Structural and multidisciplinary optimization*, 37(2):131–147, 2008.

[96] S.-J. Wu and P.-T. Chow. Genetic algorithms for nonlinear mixed discrete-integer optimization problems via meta-genetic parameter optimization. *Engineering Optimization*, 24(2):137–159, 1995.

[97] Z. Yuan, M. Montes de Oca, M. Birattari, and T. Stützle. Modern Continuous Optimization Algorithms for Tuning Real and Integer Algorithm Parameters. In M. Dorigo et al., editors, *Proceedings of ANTS 2010, the Seventh International Conference on Swarm Intelligence*, volume 6234 of *LNCS*, pages 204–215. Springer-Verlag, Berlin, Germany, 2010.

[98] E. Zahara and Y.T. Kao. Hybrid Nelder-Mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Systems with Applications*, 36(2):3880–3886, 2009.