Ant Colony Optimization for Multi-objective Optimization Problems

Inès Alaya
SOIE/LIRIS, National School of Computer Sciences
Manouba Campus 2010
ines.alaya@isg.rnu.tn

Christine Solnon
LIRIS, CNRS UMR 5205, Université de Lyon
43 bd 11 novembre, 69622 Villeurbanne cedex
christine.solnon@liris.cnrs.fr

Khaled Ghédira
SOIE, National School of Computer Sciences
Manouba Campus 2010
khaled.ghedira@isg.rnu.tn

Abstract

We propose in this paper a generic algorithm based on Ant Colony Optimization to solve multi-objective optimization problems. The proposed algorithm is parameterized by the number of ant colonies and the number of pheromone trails. We compare different variants of this algorithm on the multi-objective knapsack problem. We compare also the obtained results with other evolutionary algorithms from the literature.

1 Introduction

In many real-life optimization problems there are several objectives to optimize. For such multi-objective problems, there is not usually a single best solution but a set of solutions that are superior to others when considering all objectives. This set is called the *Pareto set* or *non dominated solutions*. This multiplicity of solutions is explained by the fact that objectives are generally conflicting ones. For example, when choosing a second-hand car to buy, we want it to be in good state, but also to be cheap. Therefore, a car v with price p and state s is better than a car v' with price $p' > p$ and state $s' < s$, but it is not comparable with a car v'' with price $p'' > p$ and state $s'' > s$ (or inversely, with price $p'' < p$ and state $s'' < s$).

More formally, a multi-objective optimization problem (MOP) is defined by a quadruplet (X, D, C, F) such that X is a vector of n decision variables, i.e., $X = (x_1, \ldots, x_n)$; D is a vector of n value sets defining the domains of the decision variables, i.e., $D = (d_1, \ldots, d_n)$; C is a set of constraints on X, i.e., a set of relations restricting the values that may be simultaneously assigned to the decision variables; and F is a vector of $m \geq 2$ objective functions $F(X) = (f_1(X), f_2(X), \ldots, f_m(X))$; without loss of generality, we assume that these different objective functions have to be minimized (the functions having to be maximized may be multiplied by -1).

The space of candidate solutions, noted $E(X, D, C)$, is the set of all value vectors $v \in D$ satisfying all the constraints of C. We define a partial order relation on this set as follows: a solution $v \in E(X, D, C)$ *dominates* a solution $v' \in E(X, D, C)$, noted $v \prec v'$, iff v is at least as good as v' for each of the m criteria to optimize, and strictly better than v' for at least one of these criteria, i.e., iff $\forall i \in \{1, \ldots, m\}, f_i(v) \leq f_i(v')$ and $\exists i \in \{1, \ldots, m\}, f_i(v) < f_i(v')$.

The goal of a MOP (X, D, C, F) is to find the Pareto set of all non-dominated solutions, i.e., $\{v \in E(X, D, C) | \forall v' \in E(X, D, C), \neg(v' \prec v)\}$.

In this article, we propose an approach based on Ant Colony Optimization (ACO) for solving this kind of problems. We recall in the next section the basic idea of ACO, and we describe the main features of ACO algorithms for solving multi-objective problems. Section 3 introduces a generic ACO algorithm for multiobjective problems, which is a generalization of previous proposed approaches. Section 4 introduces four different instantiations of this generic algorithm, and section 5 describes the multi-objective knapsack problem and shows how to solve this problem with our generic algorithm. Experimental results comparing the four proposed instantiations with other state-of-the-art approaches are presented in Section 6.

2 Ant Colony Optimization

The basic idea of Ant Colony Optimization (ACO) [7] is to model the problem to solve as the search for a minimum cost path in a graph, and to use artificial ants to search for good paths. The behavior of artificial ants is inspired from real ants: they lay pheromone on components (edges and/or vertices) of the graph and they choose their path with respect to probabilities that depend on pheromone trails that have been previously laid by the colony; these pheromone trails progressively decrease by evaporation. Intuitively, this indirect stigmergetic communication means aims at giving information about the quality of path components in order to attract ants, in the following iterations, towards the corresponding areas of the search space.

Artificial ants also have some extra-features that do not find their counterpart in real ants. In particular, they are usually associated with data structures that contain the memory of their previous actions, and they may apply some daemon procedures, such as local search, to improve the quality of computed paths. In many cases, pheromone is updated only after having constructed a complete path and the amount of pheromone deposited is usually a function of the quality of the complete path. Finally, the probability for an artificial ant to choose a component often depends not only on pheromone, but also on problem-specific local heuristics.

The first ant algorithm to be applied to a discrete optimization problem has been proposed by Dorigo in [6]. The problem chosen for the first experiments was the Traveling Salesman Problem and, since then, this problem has been widely used to investigate the solving capabilities of ants. The ACO metaheuristic, described e.g. in [7], is a generalization of these first ant based algorithms, and has been successfully applied to different hard combinatorial optimization problems such as quadratic assignment problems [20], vehicle routing problems [3, 9], constraint satisfaction problems [18], or maximum clique problems [19].

ACO for multiobjective problems

Recently, different papers have introduced ACO algorithms for multiobjective problems. These algorithms mainly differ with respect to the three following points.

Pheromone trails. The quantity of pheromone laying on a component represents the past experience of the colony with respect to choosing this component. When there is only one objective function, this past experience is defined with respect to this objective. However, when there are several objectives, one may consider two different strategies. A first strategy is to consider a single pheromone structure, as proposed in [14, 10, 15, 2]. In this case, the quantity of pheromone laid by ants is defined with respect to an aggregation of the different objectives. A second strategy is to consider several pheromone structures, as proposed in [13, 5, 9, 3, 4]. In this case, one usually associates a different colony of ants with each different objective, each colony having its own pheromone structure.

Solutions to reward. When updating pheromone trails, one has to decide on which of the constructed solutions laying pheromone. A first possibility is to reward solutions that find the best values for each criterion in the current cycle, as proposed in [5, 9, 4]. A second possibility is to reward every non-dominated solution of the current cycle. In this case, one may either reward all the solutions in the Pareto set, as proposed in [2] or only the new non-dominated solutions that enter in the set in the current cycle, as proposed in [13].

Definition of heuristic factors. When constructing solutions, at each step a candidate is chosen relatively to a transition probability which depends on two factors: a pheromone factor and a heuristic factor. The definition of the pheromone factor depends on the definition of the pheromone trails, as discussed in the first point. For the definition of the heuristic factor, two different strategies have been considered. A first strategy is to consider an aggregation of the different objectives into a single heuristic information, as proposed in [5, 3, 10]. A second strategy is to consider each different objective separately, as proposed in [14, 13, 9, 2, 4]. In this case, there is usually one different colony for each objective.

3 A generic ACO algorithm for multiobjective problems

In this section, we present a generic ACO framework for multi-objective problems. This algorithm will be instantiated into different variants that are described next.

The generic algorithm, called m-ACO, is implicitely parameterized by the MOP (X, D, C, F) to be solved. We shall consider that ants build solutions within a construction graph $G = (V, E)$ the definition of which depends on the problem to be solved, and that pheromone trails are associated with vertices and/or edges of this graph. We shall also assume that a heuristic information η^i is defined for every objective function $f_i \in F$.

m-ACO is also parameterized by the number of ant colonies $\#Col$ and the number of considered pheromone structures $\#\tau$. Figure 1 describes the generic framework of m-ACO($\#Col$,$\#\tau$). Basically, the algorithm follows the *MAX-MIN* Ant System scheme [20]. First, pheromone trails are initialized to a given upper bound τ_{max}. Then, at each cycle every ant constructs a solution, and pheromone trails are updated. To prevent premature convergence, pheromone trails are bounded within two given bounds τ_{min} and τ_{max} such that $0 < \tau_{min} < \tau_{max}$. The algorithm stops iterating when a maximum number of cycles has been performed.

```
**Algorithme m-ACO($\#Col$,$\#\tau$):**
    Initialize all pheromone trails to $\tau_{max}$
    repeat
        for each colony $c$ in $1..\#Col$
            for each ant $k$ in $1..nbAnts$
                construct a solution
        for $i$ in $1..\#\tau$
            update the $i^{th}$ pheromone structure trails
            if a trail is lower than $\tau_{min}$ then set it to $\tau_{min}$
            if a trail is greater than $\tau_{max}$ then set it to $\tau_{max}$
    until maximal number of cycles reached
```

Figure 1. Generic ACO algorithm for MOP

```
**Construction of a solution $S$:**
    $S \leftarrow \emptyset$
    $Cand \leftarrow V$
    while $Cand \neq \emptyset$ do
        choose $v_i \in Cand$ with probability $p_S(v_i)$
        add $v_i$ at the end of $S$
        remove from $Cand$ vertices that violate constraints
    end while
```

Figure 2. Solution construction

3.1 Solution construction

Figure 2 describes the algorithm used by ants to construct solutions in a construction graph $G = (V, E)$ the definition of which depends on the problem (X, D, C, F) to solve. At each iteration, a vertex of G is chosen within a set of candidate vertices *Cand*; it is added to the solution S and the set of candidate vertices is updated by removing vertices that violate constraints of C. The vertex v_i to be added to the solution S by ants of the colony c is randomly chosen with the probability $p^c_S(v_i)$ defined as follows:

$$p^c_S(v_i) = \frac{[\tau^c_S(v_i)]^\alpha.[\eta^c_S(v_i)]^\beta}{\sum_{v_j \in Cand}[\tau^c_S(v_j)]^\alpha.[\eta^c_S(v_j)]^\beta}$$

where $\tau^c_S(v_i)$ and $\eta^c_S(v_i)$ respectively are the pheromone and the heuristic factors of the candidate vertex v_i, and α and β are two parameters that determine their relative importance. The definition of these two factors depends on the problem to be solved and on the parameters $\#Col$ and $\#\tau$; they will be detailed later.

3.2 Pheromone update

Once all ants have constructed their solutions, pheromone trails are updated as usually in ACO algorithms: first, pheromone trails are reduced by a constant factor to simulate evaporation; then, some pheromone is laid on components of the best solution. More precisely, the quantity $\tau^i(c)$ of the i^{th} pheromone structure laying on a component c is updated as follows:

$$\tau^i(c) \longleftarrow (1-\rho) \times \tau^i(c) + \Delta\tau^i(c)$$

where ρ is the evaporation factor, such that $0 \leq \rho \leq 1$, and $\Delta\tau^i(c)$ is the amount of pheromone laid on the component c. The definition of this quantity depends on the parameters $\#Col$ and $\#\tau$; it will be detailed later.

4 Description of 4 variants of m-ACO

We now describe four different variants of our generic algorithm, that consider different numbers of colonies and pheromone structures.

4.1 Variant 1: m-ACO$_1$(m+1,m)

For this variant, the number of colonies $\#Col$ is set to $m + 1$ and the number of pheromone structures is set to m, where $m = |F|$ is the number of objectives to optimize: each colony considers a single different objective, using its own pheromone structure and heuristic information to build solutions; an extra multi-objective colony is added, that aims at optimizing all objectives.

Definition of pheromone factors. The pheromone factor $\tau^i_S(v_j)$ considered by the i^{th} single-objective colony, that aims at optimizing the i^{th} objective function f_i, is defined with respect to the i^{th} pheromone structure; depending on the considered application, it may be defined as the quantity of pheromone laying on vertex v_j or as the quantity of pheromone laying on edges between v_j and some vertices in the partial solution S.

The pheromone factor $\tau^{m+1}_S(v_j)$ considered by the extra multi-objective colony is the pheromone factor $\tau^r_S(v_j)$ of the r^{th} single-objective colony, where r is randomly chosen. So this colony considers, at each construction step, a randomly chosen objective to optimize.

Definition of heuristic factors. The heuristic factor $\eta^i_S(v_j)$ considered by the i^{th} single-objective colony, that aims at optimizing the i^{th} objective function f_i, is the i^{th} heuristic information defined by η^i.

The heuristic factor $\eta^{m+1}_S(v_j)$ considered by the extra multi-objective colony is the sum of heuristic informations associated with all objectives, i.e., $\eta^{m+1}_S(v_j) = \sum_{i=1}^{m} \eta^i_S(v_j)$.

Pheromone update. For each single-objective colony, pheromone is laid on the components of the best solution found by the i^{th} colony during the cycle, where quality of solutions is evaluated with respect to the i^{th} objective f_i only. Hence, let S^i be the solution of the i^{th} colony that minimizes f_i for the current cycle, and let S^i_{best} be the solution of the i^{th} colony that minimizes f_i over all solutions built by ants of the i^{th} colony since the beginning of the run (including the current cycle). The quantity of pheromone deposited on a solution component c for the i^{th} pheromone structure is defined as follows

$$\begin{array}{ll} \Delta\tau^i(c) = \frac{1}{(1+f_i(S^i)-f_i(S^i_{best})} & \text{if } c \text{ is a component of } S^i \\ \Delta\tau^i(c) = 0 & \text{otherwise} \end{array}$$

The multi-objective colony maintains a set of solutions: a best solution for each objective. It lays pheromone on each pheromone structure relatively to the correspondent objective with the same formulae defined for the other colonies.

4.2 Variant 2: m-ACO$_2$(m+1,m)

This second variant is very similar to the first one, and considers $m+1$ colonies and m pheromone structures: a single-objective colony is associated with every different objective, and the behavior of these single-objective colonies is defined like in variant 1; there is also an extra multi-objective colony, that aims at optimizing all objectives. The only difference between variants 1 and 2 lays in the way this multi-objective colony exploits pheromone structures of other colonies to build solutions. For this multi-objective colony, the pheromone factor $\tau_S^{m+1}(v_j)$ is defined as the sum of every pheromone factor $\tau_S^c(v_j)$ of every colony $c \in \{1, ..., m\}$, thus aggregating all pheromone information into one value.

4.3 Variant 3: m-ACO$_3$(1,1)

In this instantiation of m-ACO, there is only one colony and one pheromone structure, i.e., $\#Col = 1$ and $\#\tau = 1$.

Definition of pheromone factors. The pheromone factor $\tau_S^1(v_j)$ considered by ants of the single colony is defined with respect to the single pheromone structure; depending on the considered application, it may be defined as the quantity of pheromone laying on vertex v_j or as the quantity of pheromone laying on edges between v_j and some vertices in the partial solution S.

Definition of heuristic factors. The heuristic factor $\eta_S^1(v_j)$ considered by the single colony is the sum of heuristic informations associated with all objectives, i.e., $\eta_S^1(v_j) = \sum_{i=1}^m \eta_S^i(v_j)$.

Pheromone update. Once the colony has computed a set of solutions, every non-dominated solution (belonging to the Pareto set) is rewarded. Let S_P be this set of non-dominated solutions. The quantity of pheromone deposited on a solution component c is defined as follows

$$\begin{array}{ll} \Delta\tau^1(c) = 1 & \text{if } c \text{ is a component of a solution of } S_P \\ \Delta\tau^1(c) = 0 & \text{otherwise} \end{array}$$

Every component belonging to at least one solution of the Pareto set receives a same amount of pheromone. Indeed, these components belong to non comparable solutions.

4.4 Variant 4: m-ACO$_4$(1,m)

In this last variant, there is only one colony but m pheromone structures, i.e., $\#Col = 1$ and $\#\tau = m$.

Pheromone factor. At each step of a solution construction, ants randomly choose an objective $r \in \{1, ..., m\}$ to optimize. The pheromone factor $\tau_S^1(v_j)$ is defined as the pheromone factor associated with the randomly chosen objective r.

Heuristic factor. The heuristic factor $\eta_S^1(v_j)$ considered by the single colony is the sum of heuristic informations associated with all objectives, i.e., $\eta_S^1(v_j) = \sum_{i=1}^m \eta_S^i(v_j)$.

Pheromone update. Once the colony has computed a set of solutions, the m best solutions with respect to the m different objectives are used to reward the m pheromone structures. Let S^i be the solution of the colony that minimizes the i^{th} objective f_i for the current cycle, and let S^i_{best} be the solution that minimizes f_i over all solutions built by ants since the beginning of the run (including the current cycle). The quantity of pheromone deposited on a solution component c for the i^{th} pheromone structure is defined by

$$\begin{array}{ll} \Delta\tau^i(c) = \frac{1}{(1+f_i(S^i)-f_i(S^i_{best}))} & \text{if } c \text{ is a component of } S^i \\ \Delta\tau^i(c) = 0 & \text{otherwise} \end{array}$$

5 Application to the multi-objective knapsack problem

We show in this section how our generic algorithm m-ACO may be applied to the multi-objective knapsack problem (MOKP). This problem is one of the most studied problems in the multi-objective community. The goal is to maximize a vector of profit functions while satisfying a set of knapsack capacity constraints. More formally, an MOKP is defined as follows:

$$\begin{array}{ll} \text{maximize} & f_k = \sum_{j=1}^n p_j^k x_j, \forall k \in 1..m \\ \text{subject to} & \sum_{j=1}^n w_j^i x_j \le b_i, \forall i \in 1..q \\ & x_j \in \{0,1\}, \forall j \in 1..n \end{array}$$

where m is the number of objective functions, n the number of objects, x_j is the decision variable associated to the object j, q is the number of resource constraints, w^i_j is the quantity of resource i consumed by object j, b_i is the total available quantity of the resource i, p^k_j is the profit associated to the object j relatively to objective k.

We have studied in [1] different strategies for the definition of the pheromone structure for the multidimensional uniobjective knapsack. Indeed, to solve knapsack problems with ACO, the key point is to decide which components of the constructed solutions should be rewarded, and how to exploit these rewards when constructing new solutions. Given a solution of a knapsack problem, which is a set of selected objects $S = \{o_1, \ldots, o_k\}$, one can consider three different ways of laying pheromone trails:

- A first possibility is to lay pheromone trails on each object selected in S. In this case, the idea is to increase the desirability of each object of S so that, when constructing a new solution, these objects will be more likely to be selected.
- A second possibility is to lay pheromone trails on each couple (o_i, o_{i+1}) of successively selected objects of S. In this case, the idea is to increase the desirability of choosing object o_{i+1} when the last selected object is o_i.
- A third possibility is to lay pheromone on all pairs (o_i, o_j) of different objects of S. In this case, the idea is to increase the desirability of choosing together two objects of S so that, when constructing a new solution S', the objects of S will be more likely to be selected if S' already contains some objects of S. More precisely, the more S' will contain objects of S, the more the other objects of S will be attractive.

Our goal in this new paper is to compare different strategies for solving multi-objective problems so that we report experiments with only one of these three different pheromone structures. We have chosen the first one, that associates a pheromone trail with every object, as it offers a good compromise between solution quality and CPU time.

The heuristic information $\eta^i_S(v_j)$ used for an objective i for a candidate object v_j is defined as follows:

$$\eta^i_S(v_j) = \frac{p_i(v_j)}{w_i(v_j)}$$

where $p_i(v_j)$ is the profit of the object v_j relatively to the objective i and $w_i(v_j)$ is the weight of the object v_j.

6 Experimental results

Experimentations are done on instances of the MOKP defined in [21]. We compare the different variants on instances with 2 objectives, 2 constraints, and 100, 250, and 500 objects respectively.

	α	β	ρ	#Ants	#Cycles
m-ACO$_1$(m+1,m)	1	4	0.1	30	100
m-ACO$_2$(m+1,m)	1	4	0.1	10	100
m-ACO$_3$(1,1)	1	8	0.01	10	3000
m-ACO$_4$(1,m)	1	4	0.01	100	3000

Table 1. Parameter settings

Parameters have been set as displayed in Table 1.

We compare also the proposed approaches with the following evolutionary algorithms: SPEA [21], FFGA [8], NSGA [17] , HLGA [11], NPGA [12], and VEGA [16].

The considered instances and the results of the different evolutionary algorithms are accessible at http://www.tik.ee.ethz.ch/∼zitzler/testdata.html.

To compare performances, we use the C measure introduced in [21]: given two sets of solutions X' and X'',

$$C(X', X'') = \frac{|\{a'' \in X'' : \exists a' \in X', a' \succeq a''\}|}{|X''|}$$

When $C(X', X'') = 1$, all points in X'' are dominated by or equal to points in X', whereas when $C(X', X'') = 0$, none of the points in X'' are covered by the set X'. Note that both $C(X', X'')$ and $C(X'', X')$ have to be considered, since $C(X', X'')$ is not equal to $1 - C(X'', X')$.

6.1 Comparison of the different m-ACO variants

Table 2 compares the four m-ACO variants with the C measure. This table first shows that m-ACO$_4$(1,m) is always better than both m-ACO$_1$(m+1,m) and m-ACO$_2$(m+1,m) with respect to the C measure: some solutions of both m-ACO$_1$(m+1,m) and m-ACO$_2$(m+1,m) are covered by solutions of m-ACO$_4$(1,m) (up to 12.5% for m-ACO$_1$(m+1,m) for the instance with 500 objects), whereas no solution of m-ACO$_4$(1,m) is covered neither by solutions of m-ACO$_1$(m+1,m) nor by solutions of m-ACO$_2$(m+1,m).

When comparing m-ACO$_4$(1,m) with m-ACO$_3$(1,1), we note that no one is strictly better than the other for instances with 100 and 250 objects, as each variant has found solutions that are covered by the other variant. However, for the instance with 500 objects, m-ACO$_4$(1,m) is better than m-ACO$_3$(1,1) as 9.17% of the solutions found by m-ACO$_3$(1,1) are covered by solutions of m-ACO$_4$(1,m) whereas no solution of m-ACO$_4$(1,m) is covered by a solution of m-ACO$_3$(1,1).

Let us now compare the two multi-colony approaches m-ACO$_1$(m+1,m) and m-ACO$_2$(m+1,m). We note that no variant is better than the other for the instance with 100 ob-

	100.2			250.2			500.2		
Compared variants	**min**	**avg**	**max**	**min**	**avg**	**max**	**min**	**avg**	**max**
C(m-ACO$_4$(1,m) , m-ACO$_1$(m+1,m))	0	0.0114	0.0454	0	0.0975	0.2758	0	0.125	0.4167
C(m-ACO$_1$(m+1,m) , m-ACO$_4$(1,m))	0	0	0	0	0	0	0	0	0
C(m-ACO$_4$(1,m) , m-ACO$_2$(m+1,m))	0	0.0683	0.1463	0	0.1684	0.5	0	0.0857	0.2142
C(m-ACO$_2$(m+1,m) , m-ACO$_4$(1,m))	0	0	0	0	0	0	0	0	0
C(m-ACO$_4$(1,m) , m-ACO$_3$(1,1))	0	0.0048	0.0476	0	0.0449	0.1633	0	0.0917	0.3611
C(m-ACO$_3$(1,1) , m-ACO$_4$(1,m))	0	0.0081	0.04286	0	0.0078	0.0649	0	0	0
C(m-ACO$_1$(m+1,m) , m-ACO$_2$(m+1,m))	0	0.0219	0.0976	0.0312	0.1521	0.2812	0	0.1353	0.2353
C(m-ACO$_2$(m+1,m) , m-ACO$_1$(m+1,m))	0	0.0293	0.0811	0	0	0	0	0.0036	0.0357
C(m-ACO$_1$(m+1,m) , m-ACO$_3$(1,1))	0	0.0179	0.1026	0	0	0	0	0.0204	0.1136
C(m-ACO$_3$(1,1) , m-ACO$_1$(m+1,m))	0	0.0081	0.0429	0	0.1355	0.3333	0	0.0132	0.0714
C(m-ACO$_3$(1,1) , m-ACO$_2$(m+1,m))	0	0.0756	0.1951	0	0.1654	0.4231	0.0588	0.1176	0.1765
C(m-ACO$_2$(m+1,m) , m-ACO$_3$(1,1))	0	0	0	0	0	0	0	0.0094	0.0937

Table 2. Comparison of the 4 variants of m-ACO: each line successively displays the names of the two compared variants, and the min, average and max values of the C measure for these two variants (over 10 runs) for instances with 100, 250, and 500 objects.

jects: each variant has around 2% (on average) of its solutions that are covered by solutions of the other variant. However, for the larger instances with 250 and 500 objects, m-ACO$_1$(m+1,m) is clearly better than m-ACO$_2$(m+1,m): more than 13% of the solutions of m-ACO$_2$(m+1,m) are covered by solutions of m-ACO$_1$(m+1,m) whereas the solutions of m-ACO$_1$(m+1,m) are nearly never covered by solutions of m-ACO$_2$(m+1,m).

Finally, we also note that m-ACO$_3$(1,1) is better than m-ACO$_2$(m+1,m) whereas it is not really comparable with m-ACO$_1$(m+1,m).

6.1.1 Compromise surface

To compare variants of m-ACO, Figures 3, 4, and 5 display the compromise surfaces found by m-ACO$_1$(m+1,m), m-ACO$_3$(1,1), and m-ACO$_4$(1,m) for the instances with 100, 250, and 500 objects respectively. We do not display results for m-ACO$_2$(m+1,m) as it is covered by m-ACO$_1$(m+1,m). The Pareto surfaces displayed in these figures correspond to all the non-dominated solutions found over 10 runs.

These figures show that the non-dominated solutions found by m-ACO$_4$(1,m) are better than the ones found by m-ACO$_1$(m+1,m) and m-ACO$_3$(1,1). Indeed, the surface returned by m-ACO$_4$(1,m) is on top of the one returned by m-ACO$_1$(m+1,m) except a small portion at the extremity of the front. For the instance with 100 objects, the difference between these fronts is slight. But for the instances with 250 and 500 objects, the front returned by m-ACO$_4$(1,m) is clearly better. For the instances with 100 and 250 objects, the surface of m-ACO$_4$(1,m) is superposed in a big portion with the one of m-ACO$_3$(1,1) but it is larger.

We can conclude that the results found by comparing

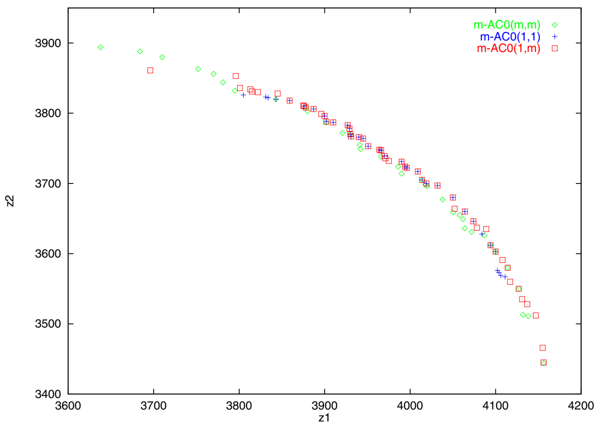

Figure 3. Instance with 100 objects.

these variants with the C measure are confirmed by the Pareto fronts displayed in this section. So, m-ACO$_4$(1,m) is generally better than the other variants especially for large instances.

6.2 Comparison of m-ACO$_4$(1,m) with evolutionary algorithms

We now compare the best variant, m-ACO$_4$(1,m), with state-of-the-art evolutionary algorithms relatively to the C measure.

Table 3 shows that m-ACO$_4$(1,m) is always better than the algorithms FFGA, NSGA, NPGA, HLGA and VEGA

Compared variants	100.2			250.2			500.2		
	min	avg	max	min	avg	max	min	avg	max
C(m-ACO$_4$(1,m) , FFGA)	0.75	0.95	1	1	1	1	1	1	1
C(FFGA , m-ACO$_4$(1,m))	0	0	0	0	0	0	0	0	0
C(m-ACO$_4$(1,m) , VEGA)	0	0.07222	0.1111	0.1052	0.51	0.7895	1	1	1
C(VEGA , m-ACO$_4$(1,m))	0	0	0	0	0	0	0	0	0
C(m-ACO$_4$(1,m) , SPEA)	0	0.02857	0.1429	0	0	0	0.0625	0.1687	0.25
C(SPEA , m-ACO$_4$(1,m))	0	0.0215	0.0741	0	0.0005	0.0159	0	0	0
C(m-ACO$_4$(1,m) , NSGA)	0	0.059	0.1026	0.1538	0.3289	0.4615	0.9	0.99	1
C(NSGA , m-ACO$_4$(1,m))	0	0	0	0	0	0	0	0	0
C(m-ACO$_4$(1,m) , HLGA)	0.15	0.25	0.35	0.3077	0.7666	1	1	1	1
C(HLGA , m-ACO$_4$(1,m))	0	0	0	0	0	0	0	0	0
C(m-ACO$_4$(1,m) , NPGA)	0.1111	0.12	0.3333	0.1667	0.7356	0.8889	1	1	1
C(NPGA , m-ACO$_4$(1,m))	0	0	0	0	0	0	0	0	0

Table 3. Comparison of m-ACO$_4$(1,m) with evolutionary algorithms: each line successively displays the names of the two compared algorithms, and the min, average and max values of the C measure for these two algorithms (over 10 runs) for instances with 100, 250, and 500 objects.

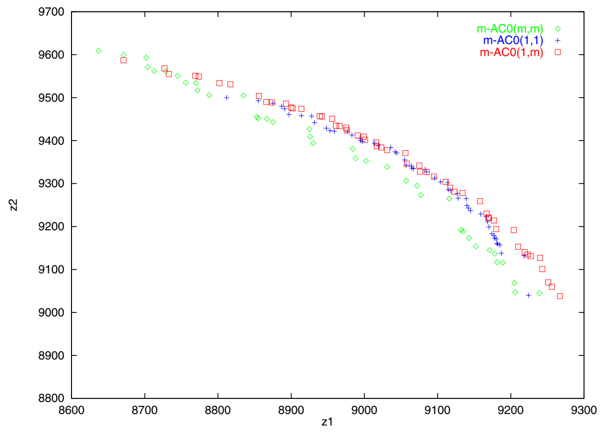

Figure 4. Instance with 250 objects.

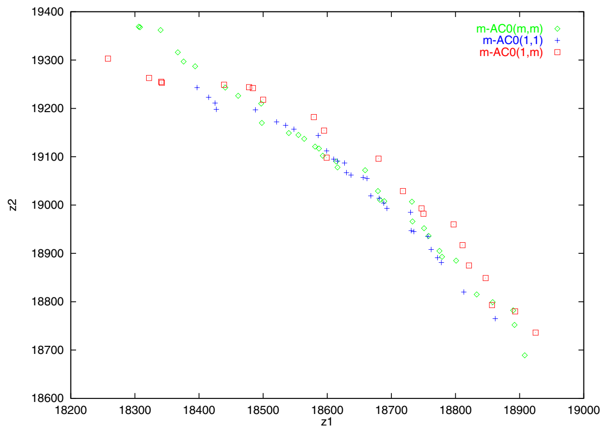

Figure 5. Instance with 500 objects.

for all the tested instances. The C measure is equal to one in several combinations. For these cases, the non dominated solutions of m-ACO$_4$(1,m) dominate all the non dominated solutions returned by the other algorithm.

When comparing m-ACO$_4$(1,m) with SPEA, we note that, for the 100 objects instance, there are non dominated solutions returned by m-ACO$_4$(1,m) that dominate others returned by SPEA, and also there are solutions from SPEA that dominate others returned by m-ACO$_4$(1,m). For the 250 objects instance, non dominated solutions of SPEA slightly dominate ones of m-ACO$_4$(1,m). But for the 500 objects instance, non dominated solutions returned by m-ACO$_4$(1,m) dominate the ones returned by SPEA. So we can conclude that m-ACO$_4$(1,m) finds better results than SPEA for large instances.

7 Conclusion

We have proposed in this paper a generic ACO algorithm for solving multiobjective optimization problems called m-ACO. This algorithm is parameterized by the number of ant colonies and the number of pheromone trails. We have tested four variants of this algorithm on the multiobjective knapsack problem (MOKP). We have found that m-ACO$_4$(1,m), where m is the number of criteria to optimize, returns globally the best results. This variant is based on new idea relatively to state-of-the-art ACO approaches. In fact, it uses a single colony with several pheromone trails. At each construction step, an ant randomly chooses a pheromone trail corresponding to an objective to optimize. The basic ideas of the other variants was used in other state-of-the-art algorithms: using a single or several colonies with respectively single or several pheromone structures. Nevertheless, these variants have some specificities. For example in m-ACO(m+1,m), there are m colonies with m pheromone structures but there is also an extra multiobjective colony that considers a randomly chosen objective at each construction step.

We have compared the variant m-ACO_4(1,m) with several evolutionary algorithms (EA) proposed in the literature for solving the MOKP. We have found that our algorithm m-ACO_4(1,m) returns generally the best results for all the considered algorithms. More precisely, m-ACO_4(1,m) finds in the Pareto set solutions that dominate others from the Pareto sets returned by the tested EA. But these algorithms, except the SPEA algorithm, don't find solutions that dominate others returned by m-ACO_4(1,m). SPEA finds better solutions than m-ACO_4(1,m) for the small instances, but for the largest instance tested m-ACO_4(1,m) finds better results.

References

[1] I. Alaya, C. Solnon, K. Ghédira, Ant algorithm for the multi-dimensional knapsack problem, *Proceedings of International Conference on Bioinspired Optimization Methods , their Applications (BIOMA 2004)*, 2004, p. 63-72.

[2] B. Barán, M. Schaerer, A Multiobjective Ant Colony System for Vehicle Routing Problem with Time Windows, *Proc. Twenty first IASTED International Conference on Applied Informatics, Insbruck, Austria*, 2003, p. 97-102.

[3] B. Bullnheimer R.F. Hartl C. Strauss, An Improved Ant system Algorithm for the Vehicule Routing Problem, *Annals of Operations Research*, vol. 89, 1999, p. 319-328.

[4] K. Doerner, R. F. Hartl, M. Teimann, Are COMPETants More Competent for Problem Solving? The Case of Full Truckload Transportation, *Central European Journal of Operations Research*, vol. 11, n2, 2003, p. 115-141.

[5] K. Doerner, W. J. Gutjahr, R. F. Hartl, C. Strauss, C. Stummer, Pareto Ant Colony Optimization: A Metaheuristic Approach to Multiobjective Portfolio Selection, *Annals of Operations Research*, 2004.

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

[7] M. Dorigo, T. Stüzle, Ant Colony Optimization, *MIT Press*, 2004.

[8] C.M. Fonseca, P.J. Fleming, Genetic algorithms for multi-objective optimization: formulation, discussion, generalization, *The Fifth International Conference on Genetic Algorithms*, 1993, p. 416-423.

[9] L. Gambardella, E.D. Taillard, G. Agazzi, MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows, *in* F. G. D. Corne, M. Dorigo (ed.), *New Ideas in Optimization*, McGraw Hill, London, UK, 1999, p. 63-76.

[10] M. Gravel, W. L. Price, C. Gagné, Scheduling Continuous Casting of Aluminium using a Multiple Objective Ant Colony Optimization Metaheuristic, *European Journal of Operational Research*, vol. 143, n1, 2002, p. 218-229.

[11] P. Hajela, C-Y. Lin, Genetic search strategies in multicriterion optimal design, *Structural Optimization*, n 4, 1992, p. 99-107.

[12] J. Horn, N. Nafpliotis, D.E. Goldberg, A niched pareto genetic algorithm for multiobjective optimization, *in* I. S. Center (ed.), *First IEEE Conference on Evolutionary Computation*, vol. 1, IEEE World Congress on Computational Computation, Piscataway, 1994, p. 82-87.

[13] S. Iredi, D. Merkle, M. Middendorf, Bi-Criterion Optimization with Multi Colony Ant Algorithms, *First International Conference on Evolutionary Multi-criterion Optimization (EMO'01)*, vol. 1993, Lecture Notes in Computer Science, 2001, p. 359-372.

[14] C. E. Mariano, E. Morales, A Multiple Objective Ant-Q Algorithm for the Design of Water Distribution Irrigation Networks, Technical Report n HC-9904, Instituto Mexicano de Tecnologa del Agua, Mexico, June, 1999.

[15] P. R. McMullen, An Ant Colony Optimization Approach to Addressing a JIT Sequencing Problem with Multiple Objectives, *Artificial Intelligence in Engineering*, vol. 15, n3, 2001, p. 309-317.

[16] J. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, *in* J. Grefenstette (ed.), *ICGA International Conference on Genetic Algorithms*, Lecture Notes in Computer Science, 1985, p. 93-100.

[17] N. Srinivas, K. Deb, Multiobjective optimization using non dominated sorting in genetic algorithms, *Evolutionary Computation*, vol. 2, 1994, p. 221-248.

[18] C. Solnon, Ants can Solve Constraint Satisfaction Problems , *IEEE Transactions on Evolutionary Computation*, vol. 6, n4, 2002, p. 347-357.

[19] C. Solnon, S. Fenet, A study of ACO capabilities for solving the Maximum Clique Problem, *Journal of Heuristics*, vol. 12, n3, 2006, p. 155-180.

[20] T. Stützle, H. Hoos, $\mathcal{MAX}-\mathcal{MIN}$ Ant System, *Journal of Future Generation Computer Systems*, vol. 16, 2000, p. 889-914.

[21] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study, the strength Pareto approach, *IEEE Transactions on Evolutionary Computation*, n 3, 1999, p. 257-271.