Using ACO in MOEA/D for Multiobjective Combinatorial Optimization

Liangjun Ke, and Qingfu Zhang, Senior Member, IEEE, and Roberto Battiti, Fellow, IEEE

Abstract—Combining ant colony optimization (ACO) and multiobjective evolutionary algorithm based on decomposition (MOEA/D), this paper proposes a multiobjective evolutionary algorithm, MOEA/D-ACO. Following other MOEA/D-like algorithms, MOEA/D-ACO decomposes an multiobjective optimization problem into a number of single objective optimization problems. Each ant (i.e. agent) is responsible for solving one subproblem. All the ants are divided into a few groups and each ant has several neighboring ants. An ant group maintains a pheromone matrix and an individual ant has a heuristic information matrix. During the search, each ant also records the best solution found so far for its subproblem. To construct a new solution, an ant combines information from its group's pheromone matrix, its own heuristic information matrix and its current solution. An ant checks the new solutions constructed by itself and its neighbors, and updates its current solution if it has found a better one in terms of its own objective. Extensive experiments have been conducted in this paper to study and compare MOEA/D-ACO with other algorithms on two set of test problems. On the multiobjective 0-1 knapsack problem, MOEA/D-ACO outperforms MOEA/D-GA on all the nine test instances. We also demonstrate that the heuristic information matrices in MOEA/D-ACO are crucial to the good performance of MOEA/D-ACO for the knapsack problem. On the biobjective traveling salesman problem, MOEA/D-ACO performs much better than Bicriterion Ant on all the 12 test instances. We also evaluate the effects of grouping, neighborhood and the location information of current solutions on the performance of MOEA/D-ACO. The work in this paper shows that reactive search optimization scheme, i.e., the “learning while optimizing” principle, is effective in improving multiobjective optimization algorithms.

Index Terms—Multiobjective optimization, ant colony optimization, multiobjective traveling salesman problem, multiobjective 0-1 knapsack problem, reactive search optimization.

I. INTRODUCTION

In many real-life applications, a decision maker (DM) has several conflicting objectives to consider and wants to determine an optimal tradeoff among them. These applications can be modeled as multiobjective optimization problems (MOP). A Pareto-optimal solution to an MOP is a candidate for the optimal tradeoff. An MOP may have many, even an infinite number of Pareto-optimal solutions. The Pareto set/front is the set of all the Pareto-optimal solutions in the decision/objective space. Without the DM’s preference information, all the Pareto-optimal solutions are equally good. The DM often wants to have a good approximation to the Pareto set and/or Pareto front (PF) available to further investigate the problem and make his final decision [1].

Many evolutionary algorithms (EA) have been developed for MOPs [2]–[9]. The major advantage of these multiobjective evolutionary algorithms (MOEA) over other methods is that they work with a population of candidate solutions and thus can produce a set of Pareto-optimal solutions to approximate the Pareto front in a single run.

Reactive search optimization (RSO) [10] [11], i.e. the “learning while optimizing” principle, has been widely accepted as a basic design principle in metaheuristics. RSO advocates the integration of machine learning techniques into heuristics and self-tunes algorithm behaviors based on online information collected during the search. Several successful evolutionary computation paradigms such as Estimation of Distribution Algorithms (EDA) and Ant Colony Optimization (ACO) adopt this principle. ACO was originally proposed for dealing with single objective combinatorial optimization problems [12]–[17]. ACO represents its knowledge learned about the problem during the search as a pheromone matrix. The pheromone value of each solution component empirically measures how likely this component is present in a promising solution. To allow for the incorporation of problem specific knowledge, ACO also uses a heuristic information matrix, which is predetermined before the search. A typical ACO employs a set of ants (i.e. agents) to search. At each generation, each ant, in a probabilistic way, constructs a solution based on the pheromone matrix and heuristic information matrix; then the pheromone values are updated by these newly constructed solutions. Encouraged by the successful applications of ACO in single objective optimization, several multiobjective ACOs (mACOs) have been developed in recent years [18]–[24]. In the design of an mACO, the following two related ingredients must be carefully considered:

- **Pheromone and heuristic information matrices:** Because the goal of mACOs is to approximate the whole Pareto front, pheromone and heuristic information matrices should contain information about possible locations of all the Pareto-optimal solutions instead of a single solution. Only one pheromone matrix and one heuristic information matrix may not serve this purpose very well, particularly when the distribution of the Pareto-optimal solutions is complicated. For this reason, most mACOs use multiple pheromone and heuristic information matrices [19], [20]. Often, each individual objective has one.
pheromone matrix and one heuristic information matrix, which can reflect the merit of each solution component for this particular objective [19], [20]. However, one pheromone matrix and one heuristic information matrix per objective are not enough for dealing with complicated MOPs. To develop an efficient mACO for approximating the whole Pareto front, it is necessary to develop an effective way to use a reasonable number of pheromone and heuristic information matrices for recording useful information.

- **Solution construction:** mACOs should be able to generate a set of diverse new solutions at each generation so that they can approximate the whole Pareto front [24]. To this end, different ants should use different combinations of pheromone and heuristic information matrices in the solution construction phase, therefore focussing onto different parts of the Pareto front. How to combine pheromone and heuristic information has not been fully explored yet.

MOEA/D (multiobjective evolutionary algorithm based on decomposition) [25] is a recent multiobjective evolutionary algorithmic framework. Like MOGLS [26]–[28], MSOPS [29] and AWA [30], it is based on conventional aggregation approaches: an MOP is decomposed into a number of single objective optimization subproblems. The objective of each subproblem is a (linearly or nonlinearly) weighted aggregation of the individual objectives. Neighborhood relations among these subproblems are defined based on the distances among their aggregation weight vectors. Each subproblem is optimized in MOEA/D by using information mainly from its neighboring subproblems. The MOEA/D framework has been studied and used for dealing with a number of multiobjective problems [31]–[42].

This paper proposes an mACO in the MOEA/D framework, called MOEA/D-ACO. In designing this algorithm, special attention has been paid to the two issues mentioned above. In MOEA/D-ACO, each ant is responsible for one single objective subproblem and it targets a particular point in the Pareto front. An ant has its own heuristic information matrix for holding its a priori information about its subproblem. Based on the neighborhood structure of the subproblems, the ants are divided into groups. Each group is to approximate a particular part of the Pareto front. The ants in the same group share one pheromone matrix which contains learned information about the position of their Pareto subregion. During the search, each ant records a current solution to its subproblem. To generate a new solution, an ant first builds its pheromone matrix by combining information from its own heuristic information matrix, the pheromone matrix of its group and its current solution. The pheromone matrix built in this way is then used to guide its solution construction. In this manner, the constructed solution is more likely to be a good solution to the ant’s subproblem. The pheromone matrix of each ant group is updated at each generation by using information extracted from the new solutions constructed by the ants in its group. Based on its own objective, each individual ant updates its current solution if it finds a better one from its neighborhood. This enables collaboration among different ant groups in MOEA/D-ACO since two neighbors may not be in the same group.

**BicriterionAnt with region update** (BicriterionAnt) [24] is probably the first mACO in which different ant groups target different subregions of the PF. The major differences between it and our proposal are: (a) Each ant in MOEA/D-ACO records a solution to its subproblem and this solution makes direct contribution to its solution construction, which, in a sense, combines the location information of an individual solution and global statistical information. Whereas BicriterionAnt has no such mechanism. (b) MOEA/D-ACO relies on neighborhood relations among individual ants for implementing collaboration among different ant groups. In contrast, different ant groups in BicriterionAnt collaborate via sharing a global solution pool. (c) An ant in MOEA/D-ACO only updates its group’s pheromone matrix, which is very easy to implement and does not involves heavy computational overhead. In BicriterionAnt, ant $i$ updates the pheromone matrix of group $k$ if ant $i$ has found a nondominated solution in the range that group $k$ is responsible for. Therefore, at each generation, BicriterionAnt has to decide which solutions are in the range of a particular ant group, which makes it hard to generalize to the case of more than two objectives [24].

The rest of this paper is organized as follows. Section II introduces multiobjective optimization, two test problems and basic decomposition approaches used in this paper. Section III presents MOEA/D-ACO. Sections IV and V compare MOEA/D-ACO with other algorithms and study the behavior of MOEA/D-ACO. Section VI concludes the paper and outline some avenues for future research.

## II. MULTIOBJECTIVE OPTIMIZATION

This section presents the general multiobjective optimization problem definition and two special problems used in our experiments.

### A. Problem Definition

A general multiobjective optimization problem (MOP) can be stated as follows:

\[
\begin{align*}
\text{minimize} & \quad F(x) = (f_1(x), \ldots, f_m(x)) \\
\text{subject to} & \quad x \in \Omega
\end{align*}
\]

where $\Omega$ is the decision (variable) space, $F : \Omega \rightarrow \mathbb{R}^m$ consists of $m$ real-valued objective functions and $\mathbb{R}^m$ is called the objective space. The attainable objective set is defined as the set $\{F(x) | x \in \Omega\}$.

Let $u, v \in \mathbb{R}^m$, $u$ is said to dominate $v$ if and only if $u_i \leq v_i$ for every $i \in \{1, \ldots, m\}$ and $u_i < v_i$ for at least one index $j \in \{1, \ldots, m\}$. A point $x^* \in \Omega$ is Pareto-optimal if there is no point $x \in \Omega$ so that $F(x)$ dominates $F(x^*)$. $F(x^*)$ is then called a Pareto-optimal (objective) vector. In other words, any improvement in one objective of a Pareto-optimal point must lead to deterioration to at least another objective. The set of

\[^1\text{This definition of domination is for minimization. All the inequalities should be reversed if the goal is to maximize the objectives in (1).}\]
all the Pareto-optimal points is called the Pareto set (PS) and the set of all the Pareto-optimal objective vectors is called the Pareto front (PF) [1].

The following two MOPs are used in this paper as benchmarks.

1) Multiobjective 0-1 Knapsack Problem (MOKP): Given a set of \( n \) items and a set of \( m \) knapsacks, the MOKP can be stated as:

\[
\text{maximize } f_i(x) = \sum_{j=1}^{n} p_{i,j} x_j, \quad i = 1, \ldots, m
\]

subject to

\[
\sum_{j=1}^{n} w_{i,j} x_j \leq c_i, \quad i = 1, \ldots, m
\]

where \( p_{i,j} \geq 0 \) is the profit of item \( j \) in knapsack \( i \), \( w_{i,j} \geq 0 \) is the weight of item \( j \) in knapsack \( i \), and \( c_i \) is the capacity of knapsack \( i \). \( x_i = 1 \) means that item \( i \) is selected and put into all knapsacks.

The MOKP is NP-hard and can model a variety of applications in resource allocation. A set of nine test instances of the above problem have been proposed in [3] and widely used in testing multiobjective heuristics.

2) Multiobjective Traveling Salesman Problem (MTSP): Given a set of \( n \) cities and \( m \) symmetric matrices \( (c_{j,k})_{n \times n} \), \( i = 1, \ldots, m \), where \( c_{j,k} > 0 \) can be interpreted as the \( i \)-th cost of traveling between city \( j \) and \( k \). The problem is as follows:

\[
\text{minimize } f_i(x) = \sum_{j=1}^{n-1} c_{x_j,x_{j+1}} + c_{x_1,x_n} \quad i = 1, \ldots, m
\]

where \( x \) is a permutation of \( n \) cities, representing a tour that visits each city exactly once. The MTSP is NP-hard too. In real-world applications, the different costs can be length, gasoline consumption, time, risk, etc.

### B. MOP Decomposition

Several approaches have been proposed for decomposing an MOP into a number of single objective optimization subproblems [1], [43]-[45]. In the following, we introduce two most commonly-used approaches.

1) The Weighted Sum Approach: Let \( \lambda = (\lambda_1, \ldots, \lambda_m) \) be a weight vector, i.e., \( \sum_{i=1}^{m} \lambda_i = 1 \) and \( \lambda_i \geq 0 \) for all \( i = 1, \ldots, m \). Then the optimal solutions to the following single optimization problems:

\[
\text{minimize } g^w(x|\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x)
\]

are Pareto-optimal to (1) if the PF of (1) is convex, where we use \( g^w(x|\lambda) \) to emphasize that \( \lambda \) is a weight vector in this objective function, while \( x \) is the variable to optimize. However, when the PS is not convex, the weighted sum approach may not be able to find some Pareto-optimal solutions.

2) The Tchebycheff Method Approach: the single objective functions to be minimized are in the form

\[
g^k(x|\lambda) = \max_{1 \leq i \leq m} \{ \lambda_i (f_i(x) - z^*_i) \}
\]

where \( z^* = (z^*_1, \ldots, z^*_m) \) is the reference point, i.e.,

\[
z^*_i = \min_{x \in \Omega} f_i(x)
\]

for each \( i = 1, \ldots, m \). Under some mild condition [1], for each Pareto-optimal point \( x^* \) there exists a weight vector \( \lambda \) such that \( x^* \) is the optimal solution of (5) and each optimal solution of (5) is Pareto-optimal to (1).

To obtain a set of different Pareto-optimal solutions of (1), one can solve a set of single objective optimization problems with different weight vectors defined by (4), (5) or any other decomposition approaches.

### III. THE NEW PROPOSAL: MOEA/D-ACO

This section describes the MOEA/D-ACO and its implementations for MOKP and MTSP.

#### A. Algorithmic Framework

MOEA/D-ACO first decomposes the MOP into \( N \) single objective subproblems by choosing \( N \) weight vectors \( \lambda^1, \ldots, \lambda^N \). Subproblem \( i \) is associated with weight vector \( \lambda^i \) and its objective function is denoted as \( g(x|\lambda^i) \). MOEA/D-ACO employs \( N \) ants for solving these single objective subproblems. Ant \( i \) is responsible for subproblem \( i \).

Because \( g(x|\lambda) \) is a continuous function of \( \lambda \), two subproblems are likely to have similar solutions if their weight vectors are close. Motivated by this observation, MOEA/D has the following concept:

- **Neighborhood:** the neighborhood \( B(i) \) of ant \( i \) contains \( T \) ants whose subproblem’s weight vectors are the \( T \) closest to \( \lambda^i \) among all the \( N \) weight vectors. We always assume that \( i \in B(i) \), in other words, ant \( i \) is its own neighbor.

In addition, MOEA/D-ACO introduces the group concept:

- **Group:** the \( N \) ants are divided into \( K \) groups by clustering their corresponding weight vectors. Each group is intended to approximate a small range of the PF.

Fig. 1 illustrates the ideas of decomposition, neighborhoods and groups in MOEA/D-ACO. To the best of our knowledge, the group concept was first used in BicriterionAnt [24] for biobjective optimization. BicriterionAnt is a Pareto-dominance based method, whereas MOEA/D-ACO adopts a decomposition approach.

At each generation, MOEA/D-ACO with \( N \) ants maintains:

- \( N \) solutions \( x^1, \ldots, x^N \in \Omega \), where \( x^i \) is the current solution to subproblem \( i \).
- \( F^1, \ldots, F^N \), where \( F^i \) is the \( F \)-value of \( x^i \), i.e., \( F^i = F(x^i) \).
- \( \tau^1, \ldots, \tau^K \), where \( \tau^j \) is the current pheromone matrix for group \( j \), storing its learned knowledge about the subregion of the PF it aims at approximating.

In the case of maximization, \( z^*_i = \max_{x \in \Omega} f_i(x) \)

In the case of maximization, “minimize” should be replaced by “maximize”.

The neighbors of ant 6

Fig. 1. In this illustrative example, the number of objectives (i.e., \(T\)) is 2, the number of ants (i.e., \(N\)) is 14, the number of groups (i.e., \(K\)) is 2, and the neighborhood size (i.e., \(T\)) is 5. The MOP is decomposed into 14 single objective subproblems. Ant \(i\) is responsible for finding the optimal solution of subproblem \(i\). Group 1 contains ant 1, 2, \ldots, 7 and group 2 contains ant 8, 9, \ldots, 14. Each ant has 5 neighbors (including itself). As shown in this figure, the neighbors of ant 6 are ant 4, 5, 6, 7 and 8.

- \(\eta^1, \ldots, \eta^N\), where \(\eta^i\) is the heuristic information matrix for subproblem \(i\), which is predetermined before the search.
- \(EP\), external archive containing all the nondominated solutions found so far.

MOEA/D-ACO works as follows:

**Step 0 Initialization:** For \(i = 1, \ldots, N\), generate initial solution \(x^i\) to subproblem \(i\) and set \(F^i = F(x^i)\). For \(i = 1, \ldots, N\), initialize \(\eta^i\), the heuristic information matrix to subproblem \(i\). For \(j = 1, \ldots, K\), initialize pheromone information matrix \(\tau^j\) for group \(j\). Initialize \(EP\) as the set of all the nondominated vectors in \(\{F^1, \ldots, F^N\}\).

**Step 1 Solution Construction:** For each \(i = 1, \ldots, N\), ant \(i\) constructs a new solution \(y^i\) by using a probabilistic rule to choose solution components. The rule is a function of \(x^i, \eta^i\) and \(\tau^j\), where ant \(i\) is in group \(j\). Compute \(F(y^i)\).

**Step 2 Update of \(EP\):** For each \(i = 1, \ldots, N\), if no vectors in \(EP\) dominate \(F(y^i)\), add \(F(y^i)\) to \(EP\) and remove from \(EP\) all the vectors dominated by \(F(y^i)\).

**Step 3 Termination:** If a problem-specific stopping condition is met, stop and output \(EP\).

**Step 4 Update of Pheromone Matrices:** For each \(j = 1, \ldots, K\), update pheromone matrix \(\tau^j\) by using information extracted from those new solutions which were constructed by ants in group \(j\) in Step 1 and have just been added to \(EP\) in Step 2.

**Step 5 Update of \(x^i\):** For \(i = 1, \ldots, N\), ant \(i\) finds \(y\), the solution with the smallest \(g(-x^i)\) value among all the new solutions which were constructed by its neighbors and have not been used to replace any other old solutions; replace \(x^i\) by \(y\) if \(g(y^{\lambda^i}) < g(x^i^{\lambda^i})\).

At each generation, each ant constructs a solution in Step 1. These newly constructed solutions update the external archive in Step 2. An ant updates the pheromone matrix of its group in Step 4 if it has found a new nondominated solution. In Step 5, each ant updates its current solution if there is a solution which (a) is better than its current solution, (b) has not been used for updating other old solutions, and (c) was generated by its neighbors. Step 3 is termination. MOEA/D-ACO is an anytime algorithm, it is always able to return complete and feasible solutions.

In the above top-level description, the details of some steps are problem-specific. In the following, we take the MOKP and MTSP as examples to show how these steps can be implemented. We would like to point out that our implementation is not unique. There are many possible ways to instantiate the above framework.

### B. MOEA/D-ACO for MOKP

1) Basic Setting:

- **Setting of \(N\) and \(\lambda^1, \ldots, \lambda^N\):** is controlled by a parameter \(H\). More precisely, \(\lambda^1, \ldots, \lambda^N\) are all the weight vectors in which each individual weight component takes a value from:

  \[
  \{ \frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H} \}.
  \]

  Therefore, the number of the weight vectors is:

  \[
  N = C_{m-1}^m. 
  \]  

2) Setting of Group: \(K\) representative weight vectors \(\xi^1, \ldots, \xi^K\) are generated by using the same method as above with a smaller value of \(H\). Then we cluster \(\lambda^1, \ldots, \lambda^N\) into \(K\) groups so that \(\xi^i\) is the closest representative vector, in terms of Euclidean distance, to all the weight vectors in group \(i\).
• **Setting of Neighborhood:** the Euclidean distance is used to compute the distance between weight vectors.

• **Decomposition Approach:** both the weighted sum and Tchebycheff approaches are used in this paper. In the Tchebycheff approach, the reference point \( z^* \) is substituted by \( z = (z_1, \ldots, z_m) \), where \( z_i \) is the best value of function \( f_i \) found so far.

• **The Data Structure of Pheromone and Heuristic Information Matrices:** because each candidate solution to the MOKP is a \( n \)-Dimensional 0-1 vector, we represent pheromone and heuristic information by using \( n \)-dimensional real valued vectors. More precisely, the pheromone matrix for group \( j \) is

\[
\tau^j = (\tau^j_1, \ldots, \tau^j_n)
\]

where \( \tau^j_k \) can be interpreted as a figure of merit, learned from the previous search, to measure the benefit of \( x_k = 1 \) in a solution for the task of group \( j \). The heuristic information matrix for ant \( i \) is

\[
\eta^i = (\eta^i_1, \ldots, \eta^i_n)
\]

where \( \eta^i_k \) is to measure the desirability, learned from the domain knowledge before the search, that \( x_k = 1 \) in a solution for subproblem \( i \).

2) **Initialization:**

- The value of the \( k \)-th element in \( \eta^i \), the heuristic information matrix for ant \( i \), is set as:

\[
\eta^i_k = \frac{\sum_{l=1}^m \lambda^i_l p_{i,l} \gamma^i_l}{\sum_{l=1}^m \gamma^i_l w_{l,k}}
\]  

(8)

where \( \gamma^i_l \) is the shadow price of constraint \( l \) in the linear programming relaxation of (4) with weight vector \( \lambda^i \). \( \eta^i_k \) is the pseudo-utility ratio [46], which has been used as a heuristic information value in [47].

- All the pheromone values are initialized to be the same large value. In our experiments:

\[
\tau^j_k = 1
\]

for all \( j = 1, \ldots, K \) and \( k = 1, \ldots, n \). The motivation is to encourage the search to focus on exploration at its early stage.

3) **Solution Construction:** Assume that ant \( i \) is in group \( j \), and that its current solution is \( x^i = (x^i_1, \ldots, x^i_n) \). In Step 1 of MOEA/D-ACO, ant \( i \) constructs a new solution \( y^i = (y^i_1, \ldots, y^i_n) \) as follows:

1. For \( k = 1, \ldots, n \), set

\[
\phi_k = (\tau^j_k + \Delta \times x^i_k)^\alpha (\eta^i_k)^\beta,
\]

(10)

where \( \Delta, \alpha, \beta > 0 \) are three control parameters. \( \phi_k \) represents the desirability that \( y^i_k = 1 \) in \( y^i \) (i.e., item \( k \) is selected) for subproblem \( i \).

2. Initialize \( y^i = (0, \ldots, 0) \) (i.e., the knapsack is empty). Set \( I = \{1, \ldots, n\} \).

Do while \( I \neq \emptyset \):

2.1 For each \( s \in I \), remove \( s \) from \( I \) if setting \( y^i_s = 1 \) (i.e., adding \( s \) to the knapsack) will make \( y^i \) infeasible.

2.2 If \( I = \emptyset \), return \( y^i \).

2.3 Generate a uniformly random number from \((0, 1)\). If it is larger than a control parameter \( r \), set \( s \) to be the index in \( I \) with the largest \( \phi \) value; otherwise, select one index \( s \) randomly from \( I \) according to the following probability:

\[
\frac{\phi_s}{\sum_{k \in I} \phi_k}.
\]

2.4 Remove \( s \) from \( I \) and set \( y^i_s = 1 \).

In (10), \( \tau^j_k \) is the pheromone trail of item \( k \) shared by all the ants in group \( j \). \( x^i_k = 1 \) means that item \( k \) is selected in the current solution of subproblem \( i \). \( x^i_k \) is private to ant \( i \). Thus \( \tau^j_k + \Delta \times x^i_k \) is a combination of the group and private knowledge. This combination can be regarded as a private pheromone trail of item \( k \) for ant \( i \). As in other ACO algorithms, \( \alpha \) and \( \beta \) are for balancing the contributions from the pheromone trails and heuristic information values.

In 2.3, the selection of \( s \) follows the so-called pseudo-random proportional rule [13]. the control parameter \( r \) is used to balance exploration and exploitation. If the generated random number is smaller than \( r \), the ant does exploitation and selects the item with the largest desirability, otherwise, it conducts random exploration biased towards items with large desirability.

4) **Update of Pheromone Matrices:** Let \( II \) be the set of all the new solutions

- that were constructed by the ants in group \( j \) in Step 1 of the current iteration,
- that were just added to EP in Step 2, and
- in which \( x_k = 1 \).

\( \tau^j_k \), the pheromone trail value of item \( k \) for group \( j \), is updated as follows

\[
\tau^j_k := \rho \tau^j_k + \frac{1}{\sum_{x \in \Pi} \sum_{1 \leq i,j \leq n} p_{i,j} - g(x|\lambda^j)}
\]

(11)

where \( \rho \) is the persistence rate of the old pheromone trails. With this update scheme, the pheromone matrix \( \tau^j \) stores some statistical information of good solutions found so far for the task of group \( j \).

To avoid abnormally large or small pheromone trails, the upper and lower bounds of pheromone values, \( \tau_{\text{max}} \) and \( \tau_{\text{min}} \), are used to limit the range of pheromone trails in our implementation.

Following MMAS [14], MOEA/D-ACO, at each generation, updates \( \tau_{\text{max}} \) as follows

\[
\tau_{\text{max}} = \frac{B + 1}{(1 - \rho)(1 - \rho) \sum_{1 \leq i,j \leq n} p_{i,j} - g_{\text{max}}}
\]

(12)

where \( B \) is the number of nondominated solutions found at the current iteration, and \( g_{\text{max}} \) is the largest value obtained for the objectives functions of all the \( N \) subproblems. \( \tau_{\text{min}} \) is always set as

\[
\tau_{\text{min}} = \varepsilon \tau_{\text{max}}
\]

(13)

where \( 0 < \varepsilon < 1 \) is a control parameter. Then for \( \tau^j_k \) obtained in (11), reset \( \tau^j_k = \tau_{\text{min}} \) if \( \tau^j_k < \tau_{\text{min}} \), and reset \( \tau^j_k = \tau_{\text{max}} \) if \( \tau^j_k > \tau_{\text{max}} \).
C. MOEA/D-ACO for MTSP

1) Basic Setting: It is the same as that for MOPK except the data structures of pheromone and heuristic information matrices, and solutions.

- Each group \( j \) has pheromone trail \( \tau_{k,l}^j \) for a link between two different city \( k \) and \( l \).
- Each ant \( i \) has heuristic information value \( \eta_{k,l}^i \) for a link between city \( k \) and \( l \).
- Each individual solution \( x \) is a tour represented by permutation.

2) Initialization: Heuristic information values are initialized as:

\[
\eta_{k,l}^i = \frac{1}{\sum_{j=1}^{m} \lambda_j c_{k,l}^i} \tag{14}
\]

Like in the implementation for MOPK, \( \tau_{k,l}^i \) is initialized to one.

3) Solution Construction: Assume that ant \( i \) is in group \( j \), and its current solution \( x^i = (x_1^i, \ldots, x_n^i) \) in Step 1 of MOEA/D-ACO, \( i \) constructs its new solution \( y^i = (y_1^i, \ldots, y_n^i) \) as follows:

1. For \( k, l = 1, \ldots, n \), set

\[
\phi_{k,l}^i = [\tau_{k,l}^i + \Delta \times \ln(x^i, \{k, l\})]^{\alpha} \eta_{k,l}^{\beta}, \tag{15}
\]

where \( \Delta, \alpha, \beta > 0 \) are three control parameters. \( \phi_{k,l}^i \) representing the attractiveness of link between city \( k \) and \( l \) to ant \( i \). The indicator function \( \ln(x^i, \{k, l\}) \) is equal to 1 if link \( \{k, l\} \) is in tour \( x^i \), 0 otherwise.

2. Ant \( i \) first randomly selects a city as its starting point and then builds its tour city by city. Suppose that its current position is \( l \) and it has not completed its tour. It chooses city \( k \) to visit next from \( C \), the set of the cities not visited so far, as follows:

\[
\text{Do while there are unvisited cities:}
\]

2.1 Generate a uniformly random number from \( (0, 1) \). If it is larger than a control parameter \( r \), choose \( k \) to be the city in \( C \) with the largest \( \phi \) value; otherwise, select \( k \) randomly from \( C \) according to the following probability:

\[
\frac{\phi_{k,l}}{\sum_{s \in C} \phi_{l,s}}.
\]

2.2 If the ant has visited all the cities, return its tour

4) Update of Pheromone Matrices: Let \( \Pi \) be the set of all the new solutions that:

- were constructed by the ants in group \( j \) in Step 1 of the current iteration,
- were just added to \( EP \) in Step 2, and
- contain the link between city \( l \) and \( k \).

\( \tau_{l,k}^j \), the pheromone trail value of link \( \{l, k\} \) for group \( j \), is updated as follows:

\[
\tau_{l,k}^j := \rho \tau_{l,k}^j + \sum_{x \in \Pi} \frac{1}{g(x|\Lambda)} \tag{16}
\]

where \( \rho \) is the persistence rate of the old pheromone trails. As in the implementation for the MOKP, \( \tau_{\text{max}} \) and \( \tau_{\text{min}} \) are used to limit the range of pheromone trails in our implementation. At each generation, \( \tau_{\text{max}} \) is set:

\[
\tau_{\text{max}} = \frac{B + 1}{(1 - \rho) g_{\text{min}}} \tag{17}
\]

where \( B \) is the number of nondominated solutions found at the current iteration, and \( g_{\text{min}} \) is the smallest value obtained for the objectives functions of all the \( N \) subproblems. \( \tau_{\text{min}} \) is:

\[
\tau_{\text{min}} = \varepsilon \tau_{\text{max}} \tag{18}
\]

where \( 0 < \varepsilon < 1 \) is a control parameter. Then for \( \tau_{l,k}^j \) obtained in (16), reset \( \tau_{l,k}^j = \tau_{\text{min}} \) if \( \tau_{l,k}^j < \tau_{\text{min}} \), and \( \tau_{l,k}^j = \tau_{\text{max}} \) if \( \tau_{l,k}^j > \tau_{\text{max}} \).

IV. COMPARISON WITH MOEA/D-GA ON MOKP

In [25], an implementation of MOEA/D with conventional genetic operators and local search (denoted by MOEA/D-GA in the following) is proposed for the MOKP. The crossover operator used is the one-point crossover and the mutation operator mutates each position of a solution with a very low probability. The local search used in MOEA/D-GA was developed by Jaszkiewicz [28].

The major reasons for this comparison are twofold: Firstly, MOEA/D-GA outperforms some other popular and efficient evolutionary algorithms such as MOGLS on the MOKP [28]. Secondly, both MOEA/D-GA and MOEA/D-ACO use the MOEA/D framework, this comparison is necessary for studying the effect of using ACO in MOEA/D.

A. Experimental Setup

The nine instances from [3] are used in our experiments.

The parameter setting in MOEA/D-GA is the same as in [25], which demonstrated that the weighted sum approach performs much better than Tchebycheff approach in MOEA/D-GA for MOKP. In our comparison, MOEA/D-GA uses the weighted sum approach as its decomposition approach.

To make a fair comparison, the parameters w.r.t the MOEA/D framework in MOEA/D-ACO are the same as in MOEA/D-GA. Both the weighted sum and Tchebycheff approaches are tested in MOEA/D-ACO. The settings of the number of ants and the number of groups for all instances are given in Table I. The other parameters are set as:

- \( T = 10 \),
- \( \alpha = 1 \),
- \( \beta = 10 \),
- \( \rho = 0.95 \),
- \( r = 0.9 \),
- \( \varepsilon = \frac{1}{2n} \), and
- \( \Delta = 0.05 \times \tau_{\text{max}} \).

Both algorithms stop after 300 generations. All the experiments have been carried out on the same computer (a Pentium IV 3GHz CPU and 2GB RAM). The programming language is C++. All the statistics are based on 30 independent runs.
MOEA/D-ACO variants are very small and therefore difficult to compare with each other. It is clear that from Table II that the final approximations obtained by MOEA/D-GA are larger than the initial IGD values obtained with heuristic information. Clearly, the larger the number of decision variables is, the larger the difference between MOEA/D-ACO and MOEA/D-GA is. Table III suggests that both MOEA/D-ACO and MOEA/D-GA consume more CPU time than MOEA/D-GA. This should be because the ACO component involves more computational overhead than crossover and mutation operators.

When concentrating the attention on function evaluations, i.e., the number of candidate solutions visited, Fig. 3-5 indicates that both MOEA/D-ACO variants are always more efficient in terms of the number of function evaluations than MOEA/D-GA in minimizing the IGD-values on all the three biobjective test instances. Actually, the final IGD values obtained by MOEA/D-GA are larger than the initial IGD values obtained by MOEA/D-ACO variants.

### B. Performance Metric

The inverted generational distance (IGD) [25] is used in assessing the performance of the algorithms in our experimental studies.

Let \( P^* \) be a set of uniformly distributed points in the objective space along the PF. Let \( P \) be an approximation to the \( PF \), the inverted generational distance from \( P^* \) to \( P \) is defined as:

\[
D(P^*, P) = \frac{\sum_{v \in P^*} d(v, P)}{|P^*|}
\]

where \( d(v, P) \) is the minimum Euclidean distance between \( v \) and the points in \( P \). If \( P^* \) is large enough to represent the PF very well, \( D(P^*, P) \) measure both the diversity and convergence of \( P \). To have a low value of \( D(P^*, P) \), \( P \) must be very close to the \( PF \) and cannot miss any part of the whole \( PF \). We chose the upper approximations obtained by Jaszkiewicz [28] as \( P^* \) in our experiments on the MOKP.

### C. Comparison Results

Table II presents the mean and standard deviation of the IGD values of the final approximations obtained by each algorithm among 30 runs for each test instance. Table III shows the average CPU time used by each algorithm. Fig. 2 plots the distribution of the final approximation with the lowest IGD value among 30 runs of each algorithm for each bi-objective test instance. MOEA/D-ACO\(^W\) and MOEA/D-ACO\(^T\) represent MOEA/D-ACO with the weighted sum approach and Tchebycheff approach, respectively. Fig. 3-5 shows the evolution of the mean of the IGD value with the number of generations.

It is clear from Table II that the final approximations obtained by both variants of MOEA/D-ACO are much better than those obtained by MOEA/D-GA in terms of the IGD metric value on all the test instances. For example, the average IGD-values in both MOEA/D-ACO variants are about 23\%, 22\%, and 25\% of those obtained by MOEA/D-GA on instances 250-2, 500-3, 750-4, respectively. From these figures, it is also clear that the differences in the final approximations between two MOEA/D-ACO variants are very small and therefore difficult to distinguish visually.

It is evident from Fig. 2 that almost all the final solutions obtained by MOEA/D-GA are dominated by those obtained by MOEA/D-ACO variants on instances 250-2, 500-2, and 750-2. Clearly, the larger the number of decision variables is, the larger the difference between MOEA/D-ACO and MOEA/D-GA is. Table III suggests that both MOEA/D-ACO variants consume more CPU time than MOEA/D-GA. This should be because the ACO component involves more computational overhead than crossover and mutation operators.

When concentrating the attention on function evaluations, i.e., the number of candidate solutions visited, Fig. 3-5 indicates that both MOEA/D-ACO variants are always more efficient in terms of the number of function evaluations than MOEA/D-GA in minimizing the IGD-values on all the three biobjective test instances. Actually, the final IGD values obtained by MOEA/D-GA are larger than the initial IGD values obtained by MOEA/D-ACO variants.

### D. More Discussion

Both MOEA/D-ACO and MOEA/D-GA adopts the same MOEA/D framework. Therefore, the success of MOEA/D-ACO should therefore be credited to its ACO part. A distinct feature of ACO is that it make good use of the heuristic information matrices and uses them to guide its solution construction. From (10), \( \beta = 0 \) in MOEA/D-ACO means that the heuristic information will make no contribution to solution construction. To investigate the effect of the heuristic information matrices on the performance of MOEA/D-ACO, we have set \( \beta = 0 \) and kept all the other parameters as the same as in Section IV.A, and tested MOEA/D-ACO on all the MOKP instances. The IGD statistics of 30 independent runs obtained by MOEA-ACO variants without heuristic information are presented and compared in Table IV. Clearly, MOEA/D-ACO variants without heuristic information performs much poorer than those with heuristic information on all the instances. For example, on instance 250-2, the average IGD-value obtained by MOEA/D-ACO\(^W\) without heuristic information is about 79 times as large as that obtained with heuristic information.

Comparing the results in Tables II and IV, one can find that MOEA/D-ACO variants without heuristic information are also poorer than MOEA/D-GA on all the instances. Thus, one can claim that the use of heuristic information significantly improves the performance of MOEA/D-ACO. As it is the case for most heuristic approaches, ACO is not a black box method, it requires intuition and intelligence of designers to be embodied into the heuristic information matrix for achieving good performances.

### V. Comparison with BicriterionAnt on MTSP

BicriterionAnt [24] is one of the best existing mACOs on the biobjective TSP [19], [22]. As pointed out in the introduction, BicriterionAnt also uses the group concept. It relies on Pareto dominance for guiding its search, whereas MOEA/D-ACO employs decomposition for dealing with MOPs. Both of them use ACO approaches for constructing new solutions. Therefore, this comparison should be useful for understanding the benefit of the decomposition approach.
TABLE II

<table>
<thead>
<tr>
<th>Instance</th>
<th>MOEA/D-ACOW</th>
<th>MOEA/D-ACOT</th>
<th>MOEA/D-GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>n  m</td>
<td>mean std dev</td>
<td>mean std dev</td>
<td>mean std dev</td>
</tr>
<tr>
<td>250 2</td>
<td>13.8 0.3 +</td>
<td>13.4 0.4 +</td>
<td>59.1 6.3</td>
</tr>
<tr>
<td>500 2</td>
<td>12.7 0.5 +</td>
<td>12.7 0.6 +</td>
<td>157.1 16.5</td>
</tr>
<tr>
<td>750 3</td>
<td>46.6 1.4 +</td>
<td>47.3 1.5 +</td>
<td>119.7 8.2</td>
</tr>
<tr>
<td>500 3</td>
<td>73.9 2.6 +</td>
<td>77.2 2.3 +</td>
<td>357.1 24.3</td>
</tr>
<tr>
<td>750 3</td>
<td>101.2 3.1 +</td>
<td>106.1 3.9 +</td>
<td>696.1 30.7</td>
</tr>
<tr>
<td>250 4</td>
<td>46.6 1.4 +</td>
<td>47.3 1.5 +</td>
<td>119.7 8.2</td>
</tr>
<tr>
<td>500 4</td>
<td>73.9 2.6 +</td>
<td>77.2 2.3 +</td>
<td>357.1 24.3</td>
</tr>
<tr>
<td>750 4</td>
<td>101.2 3.1 +</td>
<td>106.1 3.9 +</td>
<td>696.1 30.7</td>
</tr>
</tbody>
</table>

$t$-test has been made out at the 5% significance level to compare the IGD values of MOEA/D-ACO and MOEA/D-GA. “+” denotes that the mean IGD value of the corresponding algorithm are significantly smaller than that of MOEA/D-GA.

TABLE III

<table>
<thead>
<tr>
<th>Instance</th>
<th>MOEA/D-ACOW</th>
<th>MOEA/D-ACOT</th>
<th>MOEA/D-GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>n  m</td>
<td>mean std dev</td>
<td>mean std dev</td>
<td>mean std dev</td>
</tr>
<tr>
<td>250 2</td>
<td>6.4 0.0</td>
<td>6.2 0.0</td>
<td>0.9 0.0</td>
</tr>
<tr>
<td>500 2</td>
<td>10.9 0.0</td>
<td>10.7 0.0</td>
<td>2.8 0.0</td>
</tr>
<tr>
<td>750 2</td>
<td>15.7 0.0</td>
<td>15.5 0.1</td>
<td>5.8 0.1</td>
</tr>
<tr>
<td>250 3</td>
<td>15.9 0.9</td>
<td>16.2 0.0</td>
<td>2.3 0.1</td>
</tr>
<tr>
<td>500 3</td>
<td>24.6 0.1</td>
<td>24.3 0.0</td>
<td>7.2 0.2</td>
</tr>
<tr>
<td>750 3</td>
<td>33.7 0.1</td>
<td>34.0 0.1</td>
<td>13.1 0.3</td>
</tr>
<tr>
<td>250 4</td>
<td>26.1 0.0</td>
<td>25.3 0.5</td>
<td>6.4 0.1</td>
</tr>
<tr>
<td>500 4</td>
<td>43.1 0.0</td>
<td>42.3 0.0</td>
<td>16.2 0.2</td>
</tr>
<tr>
<td>750 4</td>
<td>54.6 0.1</td>
<td>55.3 0.1</td>
<td>29.4 0.4</td>
</tr>
</tbody>
</table>

TABLE IV

<table>
<thead>
<tr>
<th>Instance</th>
<th>MOEA/D-ACOW</th>
<th>MOEA/D-ACOT</th>
<th>MOEA/D-GA</th>
</tr>
</thead>
<tbody>
<tr>
<td>n  m</td>
<td>mean std dev</td>
<td>mean std dev</td>
<td>mean std dev</td>
</tr>
<tr>
<td>250 2</td>
<td>13.8 0.3</td>
<td>1010.9 137.3</td>
<td>+</td>
</tr>
<tr>
<td>500 2</td>
<td>12.7 0.5</td>
<td>3728.9 349.8</td>
<td>+</td>
</tr>
<tr>
<td>750 2</td>
<td>14.0 0.5</td>
<td>5154.7 658.7</td>
<td>+</td>
</tr>
<tr>
<td>250 3</td>
<td>46.6 1.4</td>
<td>2658.4 287.3</td>
<td>+</td>
</tr>
<tr>
<td>500 3</td>
<td>73.9 2.6</td>
<td>5830.8 865.6</td>
<td>+</td>
</tr>
<tr>
<td>750 3</td>
<td>101.2 3.1</td>
<td>873.7 38.0</td>
<td>+</td>
</tr>
<tr>
<td>250 4</td>
<td>175.4 1.5</td>
<td>2658.9 89.7</td>
<td>+</td>
</tr>
<tr>
<td>500 4</td>
<td>280.7 1.9</td>
<td>3947.8 381.8</td>
<td>+</td>
</tr>
</tbody>
</table>

$t$-test has been made out at the 5% significance level to compare the IGD values of MOEA/D-ACO using heuristic information or not. “+” denotes that the mean IGD value of the corresponding algorithm are significantly significantly larger than that of MOEA/D-ACO using heuristic information.

A. Experimental Setup

The 12 instances from [48] are used in the comparison. The parameter setting of BicriterionAnt is the same as in [22]. Specifically,
- number of groups = 3,
- number of ant in each group = 8,
- $\rho = 0.95$,
- $\alpha = 1$,
- $\gamma = 2$,
- $r = 0.9$.

The parameter settings of MOEA/D-ACO are as follows:
- total number of ants: $N = 24$,
- number of groups: $K = 3$,
- neighborhood size: $T = 10$,
- $\rho = 0.95$,
- $\alpha = 1$,
- $\gamma = 2$,
- $r = 0.9$,
- $\epsilon = \frac{1}{2\pi}$, and
- $\Delta = 0.05 \times \tau_{\text{max}}$.

In computing the IGD values, $P^*$ for each instance is the set of all the nondominated solutions found by all the runs of all the algorithms. Both algorithms stop after 3000 generations. Therefore, they construct the same number of candidate solutions for each instance. All statistics are based on 30 independent runs.
B. Experimental Results

Table V shows that the IGD values of the final approximations obtained by both MOEA/D-ACO variants are nearly one order of magnitude lower than those by BicriterionAnt on all test instances. However, there is no substantial difference between two MOEA/D-ACO variants in terms of the IGD value. From Fig. 6, the quality differences between each MOEA/D-ACO variant and BicriterionAnt can be visually detected. Clearly, the final solutions obtained by MOEA/D-ACO variants dominate almost all the final solutions found by BicriterionAnt. Fig. 7-9 suggest that both MOEA/D-ACO are more efficient in reducing the IGD values than BicriterionAnt. These experimental results indicate that the MOEA/D framework does indeed help to improve the algorithm performance. The major reason could be that the decomposition strategy makes cooperation among different ants more efficient than Pareto-dominance. It is also clear from Fig. 7-9 that MOEA/D-ACO$^T$ performs better than MOEA/D-ACO$^W$.

Table VI shows that both MOEA/D-ACO variants require much less CPU time than BicriterionAnt. This is because
BicriterionAnt needs to compare all the solutions to each other at each generation, which is very time-consuming.

C. More Discussion

Both MOEA/D-ACO and BicriterionAnt use ACO for solution construction. Unlike BicriterionAnt, each ant in MOEA/D-ACO records its current solution and uses it to compute $\phi_k$ in (15) for solution construction. MOEA/D-ACO also uses groups and neighborhoods to allow ants to exchange information. Thus, three issues arise naturally:

- What if grouping is not used in MOEA/D-ACO?
- What if the current solution makes no contribution to $\phi_k$ in (15) in MOEA/D-ACO?
- What if individual ants do not change information with their neighbors as described in Step 5 in MOEA/D-ACO?

Noting that $K = 1$ implies that no grouping is used, $\Delta = 0$ means that the current solution makes no contribution, and $T = 1$ means that an ant has no neighbor except itself, we tested MOEA/D-ACO variants in which $K = 1$, $\Delta = 0$ or $T = 1$, while the other parameters are the same as in Section V.A. The experimental results are listed in Table VII and VIII, from which we can observe:

- Both MOEA/D-ACO variants with grouping (i.e., the default parameter setting) significantly outperform their counterparts without grouping. For example, in Kroab200, the IGD value of MOEA/D-ACO with grouping is about 80% of that obtained without grouping.
- Both MOEA/D-ACO variants with the default parameter setting are significantly better than those with $\Delta = 0$ on all the 12 test instances. Taking Kroab100 as an example, the IGD value of MOEA/D-ACO with the default parameter setting is about 78% of that obtained without using location information of the current solutions.
- Both MOEA/D-ACO variants with the default parameter setting significantly outperform its counterparts with $T = 1$ on all test instances.

Therefore, one can claim that grouping, the location information of the current solutions and the neighborhood significantly improve the algorithm performance. In our experiments, there is no instance in which these techniques cause any deterioration.

VI. Conclusion and Future Research Issues

Reactive optimization search principle, i.e., “learning while optimizing”, has been widely used for designing and improving meta heuristics [11]. ACO is a successful heuristic using this principle. ACO for single objective optimization represents its problem specific knowledge as a heuristic information matrix and its knowledge learned from the previous search as
TABLE V
THE IGD STATISTICS OF THE FINAL APPROXIMATIONS OBTAINED BY MOEA/D-ACO\(^W\), MOEA/D-ACO\(^T\), AND BICRITERION\(^\text{Ant}\)

<table>
<thead>
<tr>
<th>Instance</th>
<th>MOEA/D-ACO(^W)</th>
<th>MOEA/D-ACO(^T)</th>
<th>Bicriterion(^\text{Ant})</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>mean std dev t-test</td>
<td>mean std dev t-test</td>
<td>mean std dev t-test</td>
</tr>
<tr>
<td>kroab100 100</td>
<td>2008.2 330.3 +</td>
<td>1865.7 318.6 +</td>
<td>12318.5 382.9</td>
</tr>
<tr>
<td>kroac100 100</td>
<td>1733.0 254.9 +</td>
<td>1697.8 288.4 +</td>
<td>13290.0 373.1</td>
</tr>
<tr>
<td>kroad100 100</td>
<td>1494.6 217.3 +</td>
<td>1440.2 200.8 +</td>
<td>11888.0 403.1</td>
</tr>
<tr>
<td>kroe100 100</td>
<td>1887.3 303.5 +</td>
<td>1669.6 290.3 +</td>
<td>12526.1 368.9</td>
</tr>
<tr>
<td>krobc100 100</td>
<td>1802.5 345.3 +</td>
<td>1452.9 172.8 +</td>
<td>13881.1 387.9</td>
</tr>
<tr>
<td>krobd100 100</td>
<td>1777.3 298.3 +</td>
<td>1655.1 215.7 +</td>
<td>13625.8 324.7</td>
</tr>
<tr>
<td>krobe100 100</td>
<td>2017.1 340.5 +</td>
<td>1904.6 375.7 +</td>
<td>12451.9 328.1</td>
</tr>
<tr>
<td>kroab200 200</td>
<td>4476.7 598.6 +</td>
<td>3910.1 393.8 +</td>
<td>32691.2 908.1</td>
</tr>
<tr>
<td>kroab300 300</td>
<td>8735.9 1121.2 +</td>
<td>6441.4 595.2 +</td>
<td>51776.2 1171.5</td>
</tr>
</tbody>
</table>

\(t\)-test has been made out at the 5\% significance level to compare the IGD values of MOEA/D-ACO and Bicriterion\(^\text{Ant}\). \(\ast\) denotes that the mean IGD value of the corresponding algorithm are significantly smaller than than that of Bicriterion\(^\text{Ant}\).

TABLE VI
AVERAGE RUNNING TIME (IN SECONDS) CONSUMED BY MOEA/D-ACO AND BICRITERION\(^\text{Ant}\)

<table>
<thead>
<tr>
<th>Instance</th>
<th>MOEA/D-ACO(^W)</th>
<th>MOEA/D-ACO(^T)</th>
<th>Bicriterion(^\text{Ant})</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>mean std dev t-test</td>
<td>mean std dev t-test</td>
<td>mean std dev t-test</td>
</tr>
<tr>
<td>kroab100 100</td>
<td>4.9 0.1</td>
<td>4.9 0.1</td>
<td>68.4 0.1</td>
</tr>
<tr>
<td>kroac100 100</td>
<td>4.9 0.1</td>
<td>4.9 0.1</td>
<td>68.3 0.0</td>
</tr>
<tr>
<td>kroad100 100</td>
<td>4.9 0.1</td>
<td>4.9 0.1</td>
<td>68.3 0.0</td>
</tr>
<tr>
<td>kroe100 100</td>
<td>4.9 0.1</td>
<td>4.9 0.1</td>
<td>68.3 0.0</td>
</tr>
<tr>
<td>krobc100 100</td>
<td>5.0 0.0</td>
<td>4.9 0.1</td>
<td>68.6 0.2</td>
</tr>
<tr>
<td>krobd100 100</td>
<td>4.9 0.1</td>
<td>4.9 0.1</td>
<td>68.8 0.1</td>
</tr>
<tr>
<td>krobe100 100</td>
<td>4.9 0.1</td>
<td>4.9 0.1</td>
<td>68.8 0.1</td>
</tr>
<tr>
<td>kroab200 200</td>
<td>4.9 0.1</td>
<td>4.9 0.1</td>
<td>68.5 0.1</td>
</tr>
<tr>
<td>kroab300 300</td>
<td>4.9 0.1</td>
<td>4.9 0.1</td>
<td>68.5 0.1</td>
</tr>
</tbody>
</table>

TABLE VII
THE IGD STATISTICS OF THE FINAL APPROXIMATIONS OBTAINED BY MOEA/D-ACO\(^W\) \(K = 1, \Delta = 0, \text{ and } T = 1\)

<table>
<thead>
<tr>
<th>Instance</th>
<th>default setting</th>
<th>(K = 1)</th>
<th>(\Delta = 0)</th>
<th>(T = 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>mean std dev t-test</td>
<td>mean std dev t-test</td>
<td>mean std dev t-test</td>
<td></td>
</tr>
<tr>
<td>kroab100 100</td>
<td>2008.2 330.3</td>
<td>2998.3 540.6 +</td>
<td>2471.2 294.1 +</td>
<td>2488.5 404.8 +</td>
</tr>
<tr>
<td>kroac100 100</td>
<td>1733.0 254.9</td>
<td>2916.1 477.7 +</td>
<td>2045.9 248.2 +</td>
<td>2045.1 276.6 +</td>
</tr>
<tr>
<td>kroad100 100</td>
<td>1494.6 217.3</td>
<td>2356.8 355.8 +</td>
<td>1957.4 182.8 +</td>
<td>1855.2 252.7 +</td>
</tr>
<tr>
<td>kroe100 100</td>
<td>1887.3 303.5</td>
<td>2740.2 317.4 +</td>
<td>2051.8 285.2 +</td>
<td>2042.2 329.7 +</td>
</tr>
<tr>
<td>krobc100 100</td>
<td>1802.5 345.3</td>
<td>2617.2 486.3 +</td>
<td>2011.1 230.2 +</td>
<td>2141.9 307.8 +</td>
</tr>
<tr>
<td>krobd100 100</td>
<td>1802.5 345.3</td>
<td>2601.9 407.8 +</td>
<td>2175.3 216.2 +</td>
<td>2247.6 238.5 +</td>
</tr>
<tr>
<td>krobe100 100</td>
<td>2285.9 292.8</td>
<td>3015.1 641.5 +</td>
<td>2371.3 250.4 +</td>
<td>2723.2 352.7 +</td>
</tr>
<tr>
<td>kroce100 100</td>
<td>1897.3 295.4</td>
<td>2538.3 401.9 +</td>
<td>2081.9 299.8 +</td>
<td>2216.2 344.0 +</td>
</tr>
<tr>
<td>kroce100 100</td>
<td>1852.2 312.2</td>
<td>2800.4 577.2 +</td>
<td>2134.8 218.2 +</td>
<td>2283.2 291.3 +</td>
</tr>
<tr>
<td>kroce100 100</td>
<td>2017.1 340.5</td>
<td>2355.9 554.3 +</td>
<td>2241.6 221.9 +</td>
<td>2308.8 274.0 +</td>
</tr>
<tr>
<td>kroce100 100</td>
<td>2447.6 598.6</td>
<td>5486.5 762.2 +</td>
<td>5612.1 556.3 +</td>
<td>5293.9 392.3 +</td>
</tr>
<tr>
<td>kroce100 100</td>
<td>8735.9 1121.2</td>
<td>8960.6 1534.7 +</td>
<td>9579.5 742.8 +</td>
<td>9580.3 901.8 +</td>
</tr>
</tbody>
</table>

\(t\)-test has been made out at the 5\% significance level to compare the IGD values of MOEA/D-ACO\(^W\) under different settings. \(\ast\) denotes that the mean IGD value of the corresponding algorithm are significantly larger than that of MOEA/D-ACO\(^W\) with the default setting.
a pheromone matrix. Because the goal of multiobjective optimization is to approximate the PF, instead of a single optimal solution, generalizing ACO to multiobjective optimization is not a trivial task.

MOEA/D decomposes an MOP into a number of single objective subproblems and optimizes them simultaneously. Neighborhood relations among these subproblems are defined and used to make the search efficient and effective. With the MOEA/D framework, this paper designed an mACO, MOEA/D-ACO. In this proposed algorithm, each ant is responsible for one subproblem. The ants are divided into groups and each ant has several neighboring ants. An ant group maintains a pheromone matrix and an individual ant has a heuristic information matrix. During the search, each ant also records the best solution found so far for its subproblem. To construct a new solution, an ant combines information from its group’s pheromone matrix, its own heuristic information matrix and its current solution.

We conducted extensive experimental results on two paradigmatic MOPs. The experimental results on the multiobjective 0-1 knapsack problem show that MOEA/D-ACO outperforms MOEA/D-GA on all the nine test instances. Furthermore, the heuristic information matrices in MOEA/D-ACO make a very significant contribution to its performance. In other words, MOEA/D-ACO successfully inherits the learning and adaptation ability of single objective ACO. Comparison of MOEA/D-ACO with BicriterionAnt, a very efficient mACO proposed in the previous literature, on the biobjective TSP has shown that MOEA/D-ACO performs much better on all the 12 test instances. We also studied the effects of grouping, the neighborhood and the use of location information of current solutions on the performance of MOEA/D-ACO.

We believe that our proposed hybrid approach can be extended in several ways. The future research avenues include:

- studies of MOEA/D-ACO on other hard or real-world multiobjective problems. Specific attention should be paid for how to decompose these MOPs and how to define their heuristic and pheromone matrices.
- investigation of online parameter-tuning and configuration techniques in MOEA/D-ACO. Although the manual tuning in this paper provided excellent results, it is worthwhile studying automatic tuning techniques based on machine learning for extending MOEA/D-ACO to different problems [11] and [49].
- combinations of MOEA/D-ACO with single objective search techniques. Although MOEA/D framework provides a very natural framework for using single objective search techniques, it is worthwhile studying when and
how a single objective search techniques can be used in MOEA/D-ACO.
• use of MOEA/D-ACO in two-phase Pareto search. A two-phase strategy first generates a set of high quality solutions by a search method, and then applies Pareto local search on them to generate approximate Pareto-optimal solutions [50]. It is very interesting to study how to use MOEA/D-ACO in the first phase.
• use of the ideas in MOEA/D-ACO for generating EDAs to multiobjective optimization. Hybrids of EDAs with other techniques have been widely studied and used for single objective optimization (e.g., [51], [52]). These experiences should be very useful for combine in MOEA/D with EDA.
• studies of combination of MOEA/D-ACO with the DM’s preference information. The Brain-Computer Evolutionary Multiobjective Optimization scheme (BC-EMO) approximates an utility function in an interactive manner [53] for reducing the cognitive burden on the DM. It is very interesting to study how the DM’s preference can
be integrated into MOEA/D-ACO in such a way.

The C++ source code of MOEA/D-ACO can be downloaded from Qingfu Zhang’s homepage:
http://dces.essex.ac.uk/staff/qzhang/.

REFERENCES
Fig. 9. The evolution of average IGD values with the number of generations in kroa300.


[53] R. Battiti and A. Passerini, “Brain-computer evolutionary multi-objective
optimization (BC-EMO): a genetic algorithm adapting to the decision
maker,” *IEEE Transactions on Evolutionary Computation*, 2010, in
press.