Hybridization of Decomposition and Local Search for Multiobjective Optimization

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

***Abstract*—Combining ideas from evolutionary algorithms, decomposition approaches, and Pareto local search, this paper suggests a simple yet efficient memetic algorithm for combinatorial multiobjective optimization problems: memetic algorithm based on decomposition (MOMAD). It decomposes a combinatorial multiobjective problem into a number of single objective optimization problems using an aggregation method. MOMAD evolves three populations: 1) population P_L for recording the current solution to each subproblem; 2) population P_P for storing starting solutions for Pareto local search; and 3) an external population P_E for maintaining all the nondominated solutions found so far during the search. A problem-specific single objective heuristic can be applied to these subproblems to initialize the three populations. At each generation, a Pareto local search method is first applied to search a neighborhood of each solution in P_P to update P_L and P_E. Then a single objective local search is applied to each perturbed solution in P_L for improving P_L and P_E, and reinitializing P_P. The procedure is repeated until a stopping condition is met. MOMAD provides a generic hybrid multiobjective algorithmic framework in which problem specific knowledge, well developed single objective local search and heuristics and Pareto local search methods can be hybridized. It is a population based iterative method and thus an anytime algorithm. Extensive experiments have been conducted in this paper to study MOMAD and compare it with some other state-of-the-art algorithms on the multiobjective traveling salesman problem and the multiobjective knapsack problem. The experimental results show that our proposed algorithm outperforms or performs similarly to the best so far heuristics on these two problems.**

***Index Terms*—Decomposition, local search, multiobjective knapsack problem, multiobjective optimization, multiobjective traveling salesman problem.**

I. Introduction

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

$$\begin{aligned} &\text{maximize} \quad F(x) = (f_1(x), \ldots, f_m(x)) \\ &\text{subject to} \quad x \in \Omega \end{aligned} \tag{1}$$

Manuscript received May 8, 2013; revised December 11, 2013; accepted December 17, 2013. Date of publication January 10, 2014; date of current version September 12, 2014. This work was supported in part by the Open Research Fund of the State Key Laboratory of Astronautic Dynamics under Grant 2013ADL-DW0403, and in part by the National Natural Science Foundation of China under Grant 61203350, 61173109, 61175063. This paper was recommended by Associate Editor H. Ishibuchi.

L. Ke is with the State Key Laboratory for Manufacturing Systems Engineering, Xian Jiaotong University, Xi'an 710049, China (e-mail: keljxjtu@xjtu.edu.cn).

Q. Zhang is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: qingfu.zhang@cityu.edu.hk). He is on leave from the School of Computer Science and Electronic Engineering, University of Essex, Colchester CO4 3SQ, U.K. (e-mail: qzhang@essex.ac.uk).

R. Battiti is with the Department of Information Engineering and Computer Science, University of Trento, Trento 38122, Italy (e-mail: battiti@disi.unitn.it).

Digital Object Identifier 10.1109/TCYB.2013.2295886

where Ω is the feasible region in the decision space, $F : \Omega \to R^m$ consists of m real-valued objective functions. The attainable objective set is $\{F(x)|x \in \Omega\}$.

Let $u, v \in R^m$, u is said to dominate v, denoted by $u \succ v$, if and only if $u_i \geq v_i$ for every $i \in \{1, \ldots, m\}$ and $u_j > v_j$ for at least one index $j \in \{1, \ldots, m\}$.[1] Given a set S in R^m, a point in it is called nondominated in S if no other point in S can dominate it. A point $x^* \in \Omega$ is Pareto-optimal if it is nondominated in the attainable objective set. $F(x^*)$ is then called a Pareto-optimal (objective) vector. In other words, any improvement in one objective of a Pareto optimal point must lead to a deterioration of at least another objective. The set of all the Pareto-optimal points is called the Pareto set (PS) and the set of all the Pareto-optimal objective vectors is the Pareto front (PF) [1]. In many real life applications, the PF is of great interest to decision makers for understanding the tradeoff nature of different objectives and selecting their final solutions.

If Ω is a finite set, then (1) is called a combinatorial MOP. Finding the exact PF of a real world combinatorial MOP is often NP-hard by nature even though its single objective counterpart is not [2]. For this reason, heuristics, particularly, hybrid heuristics, have been widely used and studied for approximating the PFs of combinatorial MOPs. Elements and ideas from many different single objective heuristics such as variable neighborhood search [3], tabu search [4], iterative local search [5], guided local search [6], simulated annealing [7], ant colony optimization [8], and reactive search optimization [9] have been generalized to combinatorial multiobjective optimization. In most hybrid heuristics developed over the last two decades, the following three highly related basic techniques are frequently employed.

1) Pareto local search (PLS): It is a natural extension of single objective local search methods [10]–[12]. PLS works with a set of mutually nondominated solutions. It explores some or all of the neighbors of these solutions to find new solutions for updating this set at each iteration. Several variants of PLS have been proposed and investigated on various problems (e.g., [13], [14]).
2) Aggregation: An MOP can be transformed into a single objective optimization problem by linearly or nonlinearly aggregating all the individual objectives $f_1, \ldots, f_m$ into one single objective. Under mild conditions, an optimal solution to this single objective

[1] In the case of minimization, the inequality signs should be reversed.

problem is Pareto-optimal to (1). Thus, a single objective optimization method can be used for finding a set of Pareto optimal solutions by solving a set of such single objective problems with different weights [2].

3) Multiobjective evolutionary algorithms (MOEA): A typical MOEA evolves a working population (i.e., on-line population) of a prefixed size by using selection and reproduction operators. An MOEA selection operator can be based on Pareto dominance (e.g., [15]–[17]), aggregation (e.g., [18]–[22]) or some performance indicators (e.g. [23]). A second population (i.e., archive), is often employed in MOEAs to store nondominated solutions generated during their evolutionary process [22].

One of the most successful and simplest hybrid multiobjective heuristic frameworks is the two phase Pareto local search (2PPLS) [12]. It combines PLS and aggregation. As its name suggests, 2PPLS consists of two phases. Phase 1 generates an approximation to the set of all the efficient supported solutions (their definitions will be given in Section II) by solving a number of linear aggregation problems. Phase 2 applies a PLS to every solution generated in Phase 1 to find non-supported Pareto optimal solutions. Since it has only two phases, it is necessary that the initial solutions provided by Phase 1 are of high quality and as many as possible. Therefore, as pointed out in [12], Phase 1 often incurs very heavy computational overhead. For this reason, 2PPLS is not very suitable for dealing with more than three objectives. Another disadvantage is that it has no mechanism to escape from locally optimal solutions and it may stop when there is still computational time available. Arguably, a very natural way for overcoming these disadvantages is to implement the 2PPLS idea in an iterative framework, which makes it possible for the search to use any given amount of computational time and to implement a mechanism for jumping out of attraction basins of local optima.

The multiobjective evolutionary algorithm based on decomposition (MOEA/D) [22] uses aggregation to decompose an MOP into a number of single objective subproblems. Solutions to these different subproblems are evolved and improved in a collaborative way. Single objective optimization methods can be easily employed in the MOEA/D framework. Since MOEA/D is an iterative algorithm, any amount of computational time can be, at least in principle, used to improve the quality of its solutions. The MOEA/D algorithmic framework has been adopted in a number of MOEAs (e.g., [24]–[36]).

This paper combines the ideas from 2PPLS and MOEA/D and proposes a Multiobjective MOMAD. It is simple yet efficient and makes use of all the three basic techniques mentioned earlier in this section. MOMAD maintains three populations: Each solution in the first population (denoted by P_L in this paper) is associated with a weighted sum aggregation subproblem. The second population (denoted by P_P) stores some very promising solutions selected from P_L. The third population P_E is an archive for recording all the nondominated solutions found so far. A problem-specific single objective heuristic can be applied to these subproblems to initialize the three populations. At each generation, a Pareto local search method is first applied to search a neighborhood of each solution in P_P to update P_L and P_E. Then a single objective local search is applied to each perturbed solution in P_L for improving P_L and P_E, and reinitializing P_P. The procedure is repeated until a stopping condition is met. Extensive experiments have been conducted in this paper to study MOMAD and compare it with some other state-of-the-art algorithms on the multiobjective traveling salesman problem (mTSP) and the multiobjective knapsack problem (mKP). The experimental results show that our proposed algorithm outperforms or performs similarly to the best so far heuristics on these two problems.

The rest of the paper is organized as follows. Section II presents all the details of our proposed MOMAD. Section III gives an implementation of MOMAD on the biobjective traveling salesman problems. Experimental studies on the effects of some control parameters on the performance of MOMAD are also conducted in this section. Section IV implements MOMAD for the multiobjective knapsack problem and compares it with four other algorithms. Section V concludes this paper and outlines some avenues for future research.

II. Main Techniques

This section introduces the major ideas and techniques used in our proposed algorithm.

A. Decomposition and Algorithmic Framework

A widely used multiobjective optimization methodology in traditional mathematical programming community is to cast an MOP into a single objective optimization problem by (linearly or nonlinearly) aggregating all the individual objectives $f_1, \ldots, f_m$ with a weight coefficient vector. To obtain an approximation to the PF, one can solve a set of such single objective problems with carefully selected weight coefficient vectors. This idea has been successfully used in some MOEAs such as MOGLS [37] and MOEA/D [22].

There are a number of aggregation approaches for decomposing an MOP into a number of single objective optimization subproblems. For simplicity, this paper uses the weighted sum approach. It considers the following single objective optimization problem:

$$\begin{array}{ll} \text{maximize} & f^{ws}(x|\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x) \\ \text{subject to} & x \in \Omega \end{array} \tag{2}$$

where $\lambda = (\lambda_1, \ldots, \lambda_m)$ is a weight vector with $\sum_{i=1}^{m} \lambda_i = 1$ and $\lambda_i \geq 0$ for all $i = 1, \ldots, m$.

For any nonnegative weight vector, (2) has an optimal solution which is Pareto optimal to (1). In the case of positive weight vectors, any optimal solutions to (2) are Pareto optimal to (1) [2]. A Pareto optimal solution is called supported efficient if it is an optimal solution to (2) with a particular weight vector [11]. Let X_{se} and PF_{se} be the set of all the supported efficient solutions and the set of their corresponding Pareto optimal objective vectors respectively, it is well-known that $PF_{se} = PF$ when PF is concave. In the case of nonconcave PFs, although $PF_{se} \neq PF$ as illustrated in Fig. 1, a basic assumption in this paper is that, in an appropriately defined neighborhood of a locally or globally optimal solution to (2), more Pareto optimal solutions can be found with a reasonable amount of computational effort. This assumption, illustrated

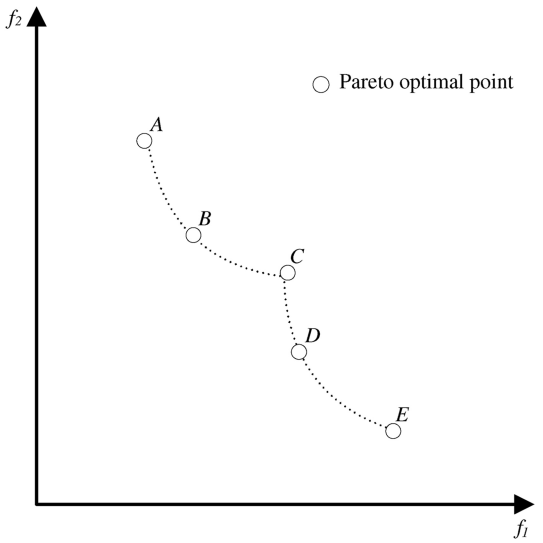

Fig. 1. Illustration of the assumption: PF is $\{A, B, C, D, E\}$, while PF_{se} is $\{A, C, E\}$. It is very likely to find D by searching a neighborhood of C. In this figure, the maximization problem is considered.

in Fig. 1, has been used in several successful metaheuristics [10] including MOGLS and 2PPLS [12].

MOMAD, the algorithm proposed in this paper, adopts the MOEA/D framework. More specifically, MOMAD first selects N weight vectors $\lambda^1, \ldots, \lambda^N$ and forms N single objective optimization subproblems. Subproblem k is defined by (2) with $\lambda = \lambda^k$. At each generation, MOMAD maintains three populations, which are as follows.

1) $P_L = \{x^1, \ldots, x^N\}$: x^k is the current solution to subproblem k. The solutions in this population will be undergone perturbation and single objective local search operators.
2) P_P: the solutions in this population will be undergone PLS.
3) P_E: it is an external population for storing all nondominated solutions found so far.

The MOMAD works as follows.

Step 0: **Initialization:**

0.1 **Initialization of P_L:** For each $k = 1, \ldots, N$, apply a single objective heuristic on subproblem k to generate solution x^k. All these solutions form P_L.

0.2 **Initialization of P_P and P_E:** Find all the nondominated solutions in P_L and let them constitute P_P. Initialize $P_E := P_P$.

Step 1 : If a preset stopping condition is met, stop and return P_E. Otherwise, go to Step 2.

Step 2 :**Pareto Local Search:** Conduct Pareto local search on P_P to update P_L and P_E. Then set $P_P =: \emptyset$ (The details of PLS can be found in Section II-B).

Step 3 : For each $k = 1, \ldots, N$

3.1 **Perturbation:** Apply a perturbation operator on x^k in P_L to generate a new solution $\tilde{x}^k$.

3.2 **Local Search:** Apply a single objective local search for subproblem k on $\tilde{x}^k$ to generate a new solution y^k.

Algorithm 1: $P_L = Update1(x', P_L)$

input : $P_L = \{x^1, \ldots, x^N\}$, x'
output: $P_L = \{x^1, \ldots, x^N\}$

```
Replacement = false
P_α := P_L
while (P_α ≠ ∅)&(Replacement = false) do
    Randomly select a solution x^k from P_α
    P_α := P_α \ {x^k}
    if f^ws(x^k|λ^k) < f^ws(x'|λ^k) then
        x^k := x', Replacement := true
    end
end
```

3.3 **Updating P_L:** $P_L = Update1(y^k, P_L)$.

3.4 **Updating P_E:** $P_E = Update2(y^k, P_E)$.

3.5 **Updating P_P:** If y^k has been added to P_E in 3.4, $P_P = Update2(y^k, P_P)$.

End of for

Go to Step 1.

In Step 0, the three populations are initialized. Since solution x^k in P_L is for single objective subproblem k, one can use a well-developed single objective heuristic to obtain the initial x^k. Then, based on Pareto dominance, it compares the solutions in P_L to find all the nondominated solutions in P_L which are then used for initializing P_E and P_P.

Step 2 conducts PLS to explore the neighborhoods of the solutions in P_P to update P_L and P_E. After this step, P_P will be set to be empty. The purpose of this step is to find those Pareto optimal solutions that may not be found by perturbation and local search in Step 3. A major computational overhead of MOMAD comes from this step.

Step 3.1 is to drive the search into a new region. The perturbation should be big enough such that the generated solution $\tilde{x}^k$ is out of the attractive basin of current x^k. Too big a perturbation, however, may take $\tilde{x}^k$ to an unpromising search area. y^k, generated in Step 3.2 by a single objective local search on $\tilde{x}^k$, could be a better solution to subproblem k or other subproblems. Steps 3.3−3.5 use y^k to update P_L, P_E and P_P. P_P contains all the new nondominated solutions generated in Step 3, and it is a subset of P_L.

In $Update1(x', P_L)$ (Algorithm 1), its inputs are x' and P_L. As mentioned earlier, each solution x^k in P_L is the current solution to subproblem k. $Update1(x', P_L)$ compares x' and solutions in P_L in a random order based on their aggregated objective function values. Due to the use of flag variable $Replacement$, x' is allowed to replace only one solution in P_L, which is useful for maintaining the diversity of P_L.

In $Update2(x', P)$ (Algorithm 2), x' is compared with every solution in P, it will be added to P if there is no solution in P which can dominate it. All the solutions in P which are dominated by x' will be removed from P.

We would like to make the following comments on the above algorithmic framework.

1) Since each solution x^i in P_L is for single objective subproblem i, single objective optimization techniques

Algorithm 2: $P = Update2(x', P)$

```
input : P, x'
output: P
if There is no solution in P which can dominate x' then
    Remove all the solutions in P which are dominated
    by x'
    Add x' to P
end
```

can be used in Step 0.1 and Step 3.2. Many effective approaches have been developed and studied in single objective combinatorial optimization for utilizing problem-specific knowledge [38]. Thus, this framework provides a natural way for using these working experiences in combinatorial multiobjective optimization.

2) In the existing two phase local search methods such as 2PPLS [12] and PD-PLS [10], each subproblem undergoes a single objective local search only once for generating one supported efficient solution. In contrast, MOMAD adopts an iterative fashion and applies a local search precedure to every subproblem many times with different starting solutions during the search. Therefore, with Pareto local search, MOMAD can explore the neighborhoods of different locally optimal solutions to each subproblem. The motivation behind the use of iteration in MOMAD is supported by the assumption illustrated in Fig. 1.
3) In some other MOEA/D variants (e.g., [39]–[41]), a neighborhood relationship among all the subproblems is defined based on the distances among their weight vectors. Correspondingly, a neighborhood relationship among all the current solutions can be established. These MOEA/D variants update a current solution mainly by an offspring of its neighboring solutions. This is based on the assumption that two neighboring subproblems have similar optimal solutions. Although this assumption is often true in combinatorial multiobjective optimization, some exceptions can be observed (e.g., [24]). For this reason, MOMAD does not employ the neighborhood concept, a new solution has a chance to replace any other current solutions in P_L.
4) A major computational overhead of MOMAD is incurred by PLS. To avoid unbearable computational cost, P_P is set to be empty at the end of Step 2. A solution is added to P_P in Step 3 only if it has entered into P_E. As a result, P_P contains all the new nondominated solutions generated in Step 3 and it is a subset of P_L. Therefore, the size of P_P cannot exceed N and neither can the number of PLS calls at each generation.

B. Pareto Local Search

MOMAD conducts PLS in Step 2. Several different versions of PLS have been developed and studied during the last several years. The PLS algorithm used in our experimental studies is a modified version of Pareto local search proposed in [12]. Its detail is given in Algorithm 3.

Algorithm 3: Pareto Local Search

```
1  input : P_P, P_L, P_E, Max
2  output: P_L, P_E
3  j := 0
4  while (P_P ≠ ∅)&(j++ < Max) do
5      P_α := ∅
6      for each x ∈ P_P do
7          for each x' ∈ N(x) do
8              update1(x', P_L)
9              if F(x) ≺ F(x') then
10                 update2(x', P_E)
11                 if x' has been added to P_E in Line 8 then
12                     update2(x', P_α)
13                 end
14             end
15         end
16     end
17     P_P := P_α
18 end
```

In this PLS algorithm, Max is the maximal number of passes in the while loop (line 2–16). It is used to bound the computational overhead. In line 1, j, the index of the current loop pass, is initialized to be 0. P_α, which is initialized to be empty in line 3, is to reset P_P at the end of each while pass in line 15. For each x' in the neighborhood of every x in P_L, $Update1(x', P_L)$ is executed in line 6 to update P_L. If x' dominates x, $update2(x', P_E)$ is executed to update P_E in Line 8. If x' has entered P_E, then line 10 executes $update2(x', P_\alpha)$ to update P_α. The two if conditions in lines 7 and 9 only allow very promising solutions to update P_E and P_L. In such a way, the computational overhead can be significantly reduced.

III. Comparison Study on Multiobjective Traveling Salesman Problem

A. Problem

The multiobjective traveling salesman problem is defined on an undirected complete graph $G = (V, E)$, where $V = \{1, \cdots, n\}$ is the set of cities (or nodes) and $E = \{e = (i, j)|i, j \in V\}$ is the set of edges. Each edge e has m distance metrics $d_{e,1}, \ldots, d_{e,m}$. Each feasible solution is a subset of edges which can form a Hamiltonian cycle. The i-th $(1 \leq i \leq m)$ objective function to be minimized is

$$f_i(x) = \sum_{e \in x} d_{e,i} \tag{3}$$

In our experiments, we use biobjective test instances collected by Lust [11],[2] which have been used to test 2PPLS.

B. Implementation of MOMAD

1) *Initialization of P_L (Step 0.1):* Each subproblem k in the mTSP is a single objective TSP with the following objective:

$$\sum_{e \in x} (\sum_{i=1}^{m} \lambda_i^k d_{e,i}). \tag{4}$$

[2]http://sites.google.com/site/thibautlust/research/.

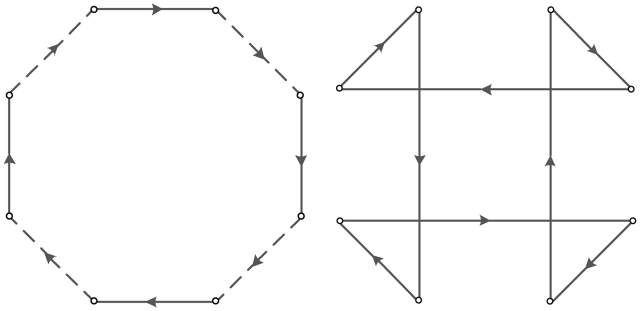

Fig. 2. Illustration of a double bridge move. (a) x^k is a solution to be perturbed. (b) $\tilde{x}^k$ is the resultant solution. This operator first assigns a direction to the circle formed by x^k, randomly removes four nonadjacent edges from the circle, and then adds four new edges to form $\tilde{x}^k$.

We use the chained Lin-Kernighan heuristic [42] on subproblem k to generate x^k in our experiments.

2) *Candidate List:* Our implementation of PLS (Step 2) and local search (Step 3.2) requires a candidate edge (CE) list. The CE list consists of all the edges of the solutions in the current P_P. It will be updated once P_P has been updated.

3) *$N(x)$ in Pareto Local Search (Step 2) and Local Search (Step 3.2) :* Given a feasible solution x, y is a neighbor of x if y is feasible and can be obtained from x in the following way: 1) remove two nonadjacent edges from x and 2) add two new edges to x to form a cycle, where at least one added edge is from the candidate list.

$N(x)$ is then the set of all the neighbors of x.

4) *Perturbation (Step 3.1):* Let x^k in P_L be the current solution to subproblem k, we apply a random double bridge move (i.e., 4-interchange [43]) on x^k to generate $\tilde{x}^k$. A double bridge move is illustrated in Fig. 2.

5) *Local Search (Step 3.2):* To conduct a local search on $\tilde{x}^k$ for subproblem k, we find the solution z from $N(\tilde{x}^k)$ (the neighborhood definition is the same as in PLS) with the highest aggregated objective function $f^{ws}(x|\lambda^k)$ value. If z is better than $\tilde{x}^k$ in terms of $f^{ws}(x|\lambda^k)$, then we replace $\tilde{x}^k$ by z and repeat this procedure, otherwise, we stop and output $\tilde{x}^k$ as y^k.

C. Performance Metrics

1) Hypervolume indicator (I_H) [17]: Let $y^* = (y_1^*, \ldots, y_m^*)$ be a point in the objective space which is dominated by any Pareto optimal objective vectors. Let S be the obtained approximation to the PF in the objective space. Then the I_H value of S (with regard to y^*) is the volume of the region which is dominated by S and dominates y^*. The higher the hypervolume value, the better the approximation.

2) Set Coverage (C-metric) [17]: Let A and B be two approximations to the PF of an MOP, $C(A, B)$ is defined as the percentage of the solutions in B that are dominated by at least one solution in A, that is

$$C(A, B) = \frac{|\{u \in B | \exists v \in A : v \text{ dominates } u\}|}{|B|} \times 100\%$$

$C(A, B)$ is not necessarily equal to $100 - C(B, A)$. $C(A, B) = 100\%$ means that all solutions in B are dominated by some solutions in A, while $C(A, B) = 0$ implies that no solution in B is dominated by any solution in A.

D. Experimental Parameter Setup

1) *Size of P_L (N) and Weight Vectors (λ) in MOMAD:* Too large N will result in very high computational cost at each generation, while too small N may make the search miss some parts of the PF. Based on these considerations, N is set as follows.

1) $N = n$ when $n = 200, 300, 400, 500$, where n is the number of cities.
2) $N = 600$ when $n = 750, 1000$.

λ^k ($k = 1, \ldots, N$) is set as

$$\lambda^k = \left(\frac{k-1}{N-1}, \frac{N-k}{N-1} \right) \tag{5}$$

2) *Max in PLS: Max* is set to be 10 in our experiments.

3) *The Stopping Condition:* The algorithm stops after 500 generations.

4) *The Number of Independent Runs:* The algorithm is run 20 times on each test instance.

E. Compared Algorithms

1) 2PPLS [12]: This algorithm is the best-so-far heuristic for the biobjective TSP instances used in our experimental studies. It consists of two phases. Its first phase generates a good approximation of the supported efficient solutions. A heuristic proposed in [44] is adopted to obtain a minimal complete set of extreme supported efficient solutions. This heuristic uses the chained Lin-Kernighan heuristic [42] to solve a set of linear aggregated TSPs. The second phase uses a PLS for generating non-supported solutions. The PLS used is the same as that described in Section II-B except it stops when no nondominated solutions can be found. The experimental results of 2PPLS used in our comparison are from its author's web[2].
2) MOEA/D-ACO [45]: It is a multiobjective ant colony algorithm using the MOEA/D framework. In this algorithm, the MOP is decomposed into a set of single objective subproblem. Each ant is responsible for one subproblem. The ants are partitioned into groups and each ant has several neighboring ants. The ants in

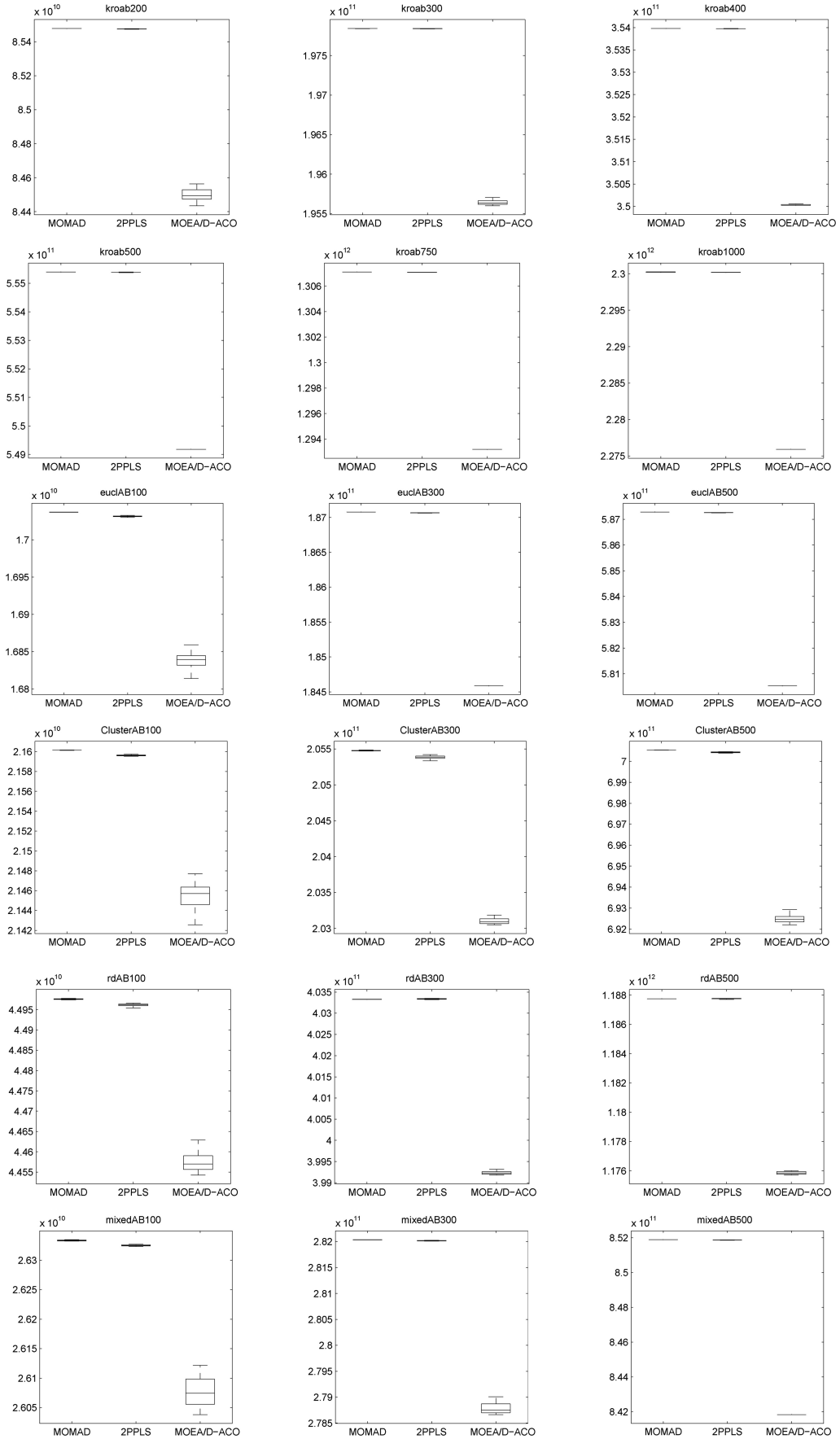

Fig. 3. Hypervolume values obtained by MOMAD, 2PPLS, and MOEA/D-ACO on the mTSP instances. In each subplot, the left is the results of MOMAD.

the same group share one pheromone matrix and each individual ant has its own heuristic information matrix. During the search, each ant 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. In the experiment, the CPU time used by this algorithm is the same as that by MOMAD, and the other parameters are set to the same values as in [45].[3]

F. Results

To compute the hypervolume values of these algorithms on each instance, y_i^* $(i = 1, \ldots, m)$ is set to be the largest value of the i-th objective among all the final solutions generated by the three compared algorithms. The hypervolume values of the final solution sets produced by these algorithms are plotted in Fig. 3. We compute the mean C-metric values of all the 20×20 comparisons on each instance and report them in Table I. The average running time consumed by each algorithm is given in Table II.

From these results, we can make the following remarks.

1) In terms of the coverage metric, MOMAD outperforms 2PPLS on all the test instances. For example, on kroab1000, the instance with 1000 cities, a solution set produced by MOMAD can on average dominate 61.2% of a solution set by 2PPLS, and 37.6% vice versa.
2) On 16 out of 18 test instances, the hypervolume values shown in Fig. 3 are consistent with the coverage metric values, which further confirms the competitiveness of MOMAD. On rdAB300 and rdAB500, the two exception instances, we have tested MOMAD with larger population sizes and found that MOMAD can perform better with regards to the hypervolume metric as shown in Fig. 4.
3) MOMAD performs much better than MOEA/D-ACO. For example, on kroab1000, a solution set produced by MOMAD can dominate 72.7% of a solution set obtained by MOEA/D-ACO on average, and 0.1% vice versa. A major reason should be that MOEA/D-ACO does not use single objective local search or Pareto local search.
4) Regarding the running time, these three algorithms were tested on different but similar computing environments. The statistics in Table II indicate that they have consumed about the same amount of CPU time on each test instance. We also want to emphasize that, unlike 2PPLS, MOMAD is an anytime algorithm. As shown in the next section, MOMAD can find better solutions if more computational resources are allocated to it.

G. More Discussions

MOMAD has two control parameters: N and Max. We have experimentally investigated the effects of the two control parameters on the algorithm performance. We have taken kroab200 as a test instance in our experimental studies and each parameter configuration has been tested on 20 independent runs.

[3]The source code is available at http://dces.essex.ac.uk/staff/zhang/

TABLE II
RUNNING TIME (IN SECONDS) OF 2PPLS, MOMAD, AND MOEA/D-ACO ON MTSP

Instance		2PPLS	MOMAD	MOEA/D-ACO
$name$	n			
kroab200	200	115	96	96
kroab300	300	232	221	221
kroab400	400	434	471	471
kroab500	500	731	819	819
kroab750	750	2301	1851	1851
kroab1000	1000	7206	3305	3305
euclAB100	100	26	21	21
euclAB300	300	259	277	277
euclAB500	500	693	708	708
ClusterAB100	100	50	47	47
ClusterAB300	300	366	345	345
ClusterAB500	500	1518	850	850
rdAB100	100	25	20	20
rdAB300	300	305	271	271
rdAB500	500	816	1213	1213
mixedAB100	100	26	22	22
mixedAB300	300	238	308	308
mixedAB500	500	866	1109	1109

2PPLS is tested on a PC with 3.0GHz CPU.
MOMAD and MOEA/D-ACO are tested on a PC with 2.4GHz CPU.

1) *Influence of the Number of Subproblems N:* N is also the size of P_L, which thus decides how many times perturbation and single objective local search operators are conducted at Steps 3.1 and 3.2 in each generation. We have tested MOMAD with $N = 100, 200, 300$, and 1700, the running time limit is set to be 600 seconds and all the other settings are the same as in the previous section. The hypervolume values of the initial P_E and the running time used for different values of N are reported in Table III and the hypervolume values of the final P_E obtained with the different running time are given in Table IV. From the experimental results, the following are evident.

1) As expected, N is a crucial factor in determining the computational overhead and performance of the initialization step in MOMAD. As shown in Table III, when $N = 50$, the running time consumed by the initialization step is 38 seconds, and the hypervolume value of initial P_E is 8.85177×10^{10}. In the case of $N = 1700$, the running time increases to 582 seconds and the hypervolume value to 8.87253×10^{10}. These results are not surprising. The larger N value is, the more times the chain Lin-Kernigham heuristic will be conducted and the more solutions will be generated in Initialization. Therefore, more running time and better P_E can be expected.
2) For all the different settings of N, the iteration steps after initialization can improve the quality of P_E effectively. For example, in the case of $N = 200$, the hypervolume value of P_E has been increased from 8.86107×10^{10} to 8.88140×10^{10}. These results should be due to two reasons. The first one is that the initial P_E is generated by the chain Lin-Kernigham heuristic, thus some of its solutions may not be globally Pareto optimal and the size of the initial P_E is also limited. As a result, there is

TABLE I
COVERAGE VALUES (%) BETWEEN MOMAD AND 2PPLS, MOEA/D-ACO ON MTSP

Instance		2PPLS		MOEA/D-ACO	
name	n	$C(A,B)$	$C(B,A)$	$C(A,B)$	$C(B,A)$
kroab200	200	33.0	13.2	75.9	0.1
kroab300	300	41.0	26.1	56.6	0.2
kroab400	400	48.8	34.5	56.4	0.1
kroab500	500	52.6	40.6	45.8	0.2
kroab750	750	62.0	36.6	64.8	0.1
kroab1000	1000	61.2	37.6	72.7	0.1
euclAB100	100	38.8	3.0	88.3	0.0
euclAB300	300	51.1	16.7	32.1	0.1
euclAB500	500	59.5	32.5	62.6	0.2
ClusterAB100	100	44.1	2.0	87.2	0.1
ClusterAB300	300	69.8	15.5	78.3	0.4
ClusterAB500	500	70.4	22.0	85.8	0.0
rdAB100	500	39.2	16.6	83.0	0.8
rdAB300	750	50.4	42.8	35.6	2.8
rdAB500	1000	57.7	40.4	43.6	1.3
mixedAB100	100	40.6	10.6	86.4	1.6
mixedAB300	300	55.1	25.3	52.9	0.9
mixedAB500	500	56.1	36.5	38.2	0.4

A corresponds to MOMAD.
B corresponds to the compared algorithm.
N.A. means unavailable.

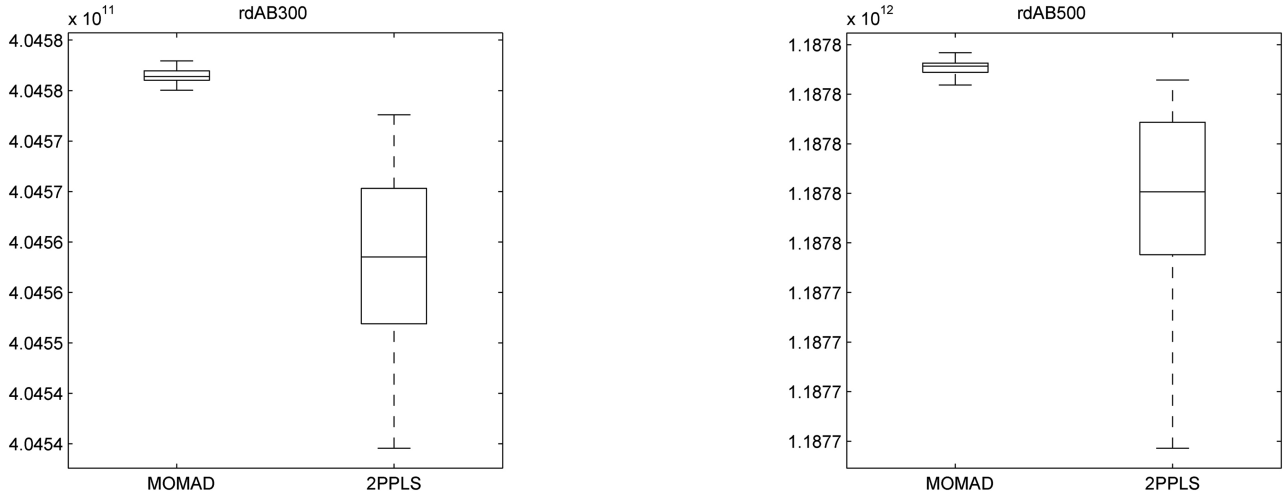

Fig. 4. Comparison of the hypervolume values obtained by 2PPLS and MOMAD with a larger N value on rdAB300 and rdAB500. $N = 400$ is for rdAB300, and 600 for rdAB500.

room for improving the initial P_E. The second reason is that perturbation and local search in Step 3.1 and Step 3.2 can explore new search areas and produce better solutions.

3) The best value of N depends on the available running time. For example, if the available running time is 100 seconds, the best value of N is 200. However, if the running time is 600s, the best value is $N = 300$.

2) *Influence of Parameter Max:* This parameter is a key parameter determining the overall computational overhead of the PLS procedure. To study its effect on the algorithm performance, we have conducted experiments with its four values $Max = 5, 10, 20$, and 40. All the other settings are the same as in Section III-D. The experimental results are reported in Table V. We can make the following comments.

1) As for the hypervolume values of the final solution sets obtained after 600 seconds, $Max = 5$ performs a bit poorer than the three other settings, and there is no big difference among $Max = 10, 20, 40$. Thus, we can claim that MOMAD is not very sensitive to the setting of Max and it is better to use a large Max value if the available running time allows one to do so.

2) In terms of the performance over the whole search period, $Max = 20, 40$ have about the same performance, $Max = 10$ shows a small edge over other values at

TABLE III
HYPERVOLUME VALUES ($\times 10^{10}$) OF THE INITIAL P_E AND THE RUNNING TIME USED FOR DIFFERENT VALUES OF N ON KROAB200

N	time	hyperpervolume
100	38	8.85177
200	73	8.86107
300	105	8.86444
1700	582	8.87253

TABLE IV
HYPERVOLUME VALUES (10^{10}) OF THE FINAL P_E OBTAINED WITH THE DIFFERENT RUNNING TIME (t) ON KROAB200

N	$t = 50$	$t = 100$	$t = 150$	$t = 300$	$t = 600$
100	8.86657	8.88108	8.88115	8.88120	8.88124
200	-	8.88129	8.88135	8.88138	8.88140
300	-	-	8.87294	8.88142	8.88151
1700	-	-	-	-	8.88139

'-' means that the initialization procedure is not completed within the given amount of CPU time.

TABLE V
INFLUENCE OF *Max* ON HYPERVOLUME (10^{10}) WHEN DIFFERENT AMOUNTS OF RUNNING TIME (t) ARE USED ON KROAB200

Max	$t = 100$	$t = 150$	$t = 300$	$t = 600$
5	8.88113	8.88130	8.88133	8.88138
10	8.88129	8.88135	8.88138	8.88140
20	8.88129	8.88132	8.88135	8.88140
40	8.88129	8.88132	8.88135	8.88140

time points $t = 150$ and 300. It may be worthwhile investigating a dynamic way for adjusting *Max* in an online manner.

IV. EXPERIMENTAL STUDIES ON MULTIOBJECTIVE 0-1 KNAPSACK PROBLEM

In the above section, the test instances are the mTSP with two objectives. Taking the multiobjective 0-1 knapsack problem (mKP) as a test bed, this section compares MOMAD with some other state-of-the-art algorithms, particularly with respect to their ability for dealing with more than two objectives.

A. mKP

Suppose there are a set of n items and m knapsacks. Each item j has m weight indexes $w_{i,j} \geq 0$ and m profit indexes $p_{i,j} \geq 0$ ($i = 1, \ldots, m$). The capacity of the knapsack for weight index i is c_i. A feasible solution x is an item subset which meets the following m constraints:

$$\sum_{j \in x} w_{i,j} \leq c_i, \ \ i = 1, \ldots, m. \tag{6}$$

The i-th objective to be maximized is

$$f_i(x) = \sum_{j \in x} p_{i,j}, \ \ i = 1, \ldots, m. \tag{7}$$

This problem is also known as the multidimensional multiobjective knapsack problem [46]. The mKP is NP-hard [2] and can model a variety of applications in resource allocation. It has also been widely used in testing multiobjective heuristics [21]. A set of nine test instances proposed in [17] are used in our experimental studies.

B. Implementation of MOMAD

1) *Initialization of P_L (Step 0.1):* x^k to subproblem k is constructed as follows. A utility value of each item j is first defined and computed

$$\sigma_j^k = \frac{\sum_{i=1}^{m} \lambda_i^k p_{i,j}}{\sum_{i=1}^{m} w_{i,j}} \tag{8}$$

where $\sum_{i=1}^{m} w_{i,j}$ is the overall weight of item k, and $\sum_{i=1}^{m} \lambda_i^k p_{i,j}$ is its weighted profit, i.e., the coefficient of x^i in the objective function of subproblem k defined by (2) in the mKP case.

We first initialize $x^k = \emptyset$, and then add items one by one to x^k until no item can be added to x^k without violating any constraint. Each item added to x^k should have the largest σ_j^k among all the items which can be added to x^k without violating any constraint.

2) *$N(x)$ in Pareto Local Search (Step 2):* If the neighborhood in PLS (Algorithm 3) is too large, the computational overhead of PLS may be very high. Our neighborhood definition is inspired by [12]. To define $N(x)$, we first define a domination relationship among all the items. An item u is said to dominate another item v if and only if

$$\frac{p_{i,u}}{\sum_{k=1}^{m} w_{k,u}} \geq \frac{p_{i,v}}{\sum_{k=1}^{m} w_{k,v}}$$

for all $i = 1, \ldots, m$, and

$$\frac{p_{i,u}}{\sum_{k=1}^{m} w_{k,u}} > \frac{p_{i,v}}{\sum_{k=1}^{m} w_{k,v}}$$

for at least one index $j \in \{1, \ldots, m\}$. Then we divide all the items into different subsets such that F_1 contains all the nondominated items in $\{1, \ldots, n\}$, F_2 contains all the nondominated items in $\{1, \ldots, n\} \setminus F_1$, and so on. The rank of the items in F_r is set to be r. The lower the rank of an item is, the better it is.

Let $U = \{1, \ldots, n\} \setminus x$ and $\theta \in (0, 1]$ We select the $\theta \times |x|$ best ranked items from set x. If there is any tie in selection, we randomly select one. Let $\bar{x}$ be the set of all these selected items. We also select $\theta \times |U|$ worst items from U and let them constitute $\bar{U}$. Roughly speaking, $\bar{x}$ consists of the worst items in x and $\bar{U}$ consists of the best items in U.

Then we define $N(x)$ as the set of all the feasible solutions which can be obtained from x by removing one item in $\bar{x}$ and adding one item from $\bar{U}$ to x. Obviously, θ determines the size of $N(x)$.

3) *Perturbation (Step 3.1):* Let x^k in P_L be the current solution to subproblem k. The output of perturbation $\tilde{x}^k$ is generated as follows: Let S be the number of items in x^k. To perturb x^k, we first remove $\lfloor \alpha \times |x^k| \rfloor$ randomly-selected items from x^k to form initial $\tilde{x}^k$, where $\alpha \in (0, 1)$ is a control parameter. Then as in Initialization of x^k in Step 0.1, add items to $\tilde{x}^k$ in a greedy way until any item addition will violate some constraint.

TABLE VI
SETTINGS OF THE NUMBER OF N IN MOMAD FOR THE MKP

Instance		N
n	m	
250	2	150
500	2	200
750	2	250
250	3	351
500	3	351
750	3	351
250	4	455
500	4	455
750	4	455

where n is the number of items and m is the number of objectives.

4) *Local Search (Step 3.2):* y is a neighbor of x if y is feasible and can be obtained from x by removing one item from it and adding a new item to it. To do local search on $\tilde{x}^k$ for subproblem k, we first find its neighboring solution z with the highest aggregated objective function $f^{ws}(x|\lambda^k)$ value. If z is better than $\tilde{x}^k$ in terms of $f^{ws}(x|\lambda^k)$, then we replace $\tilde{x}^k$ by z and repeat the neighborhood search, otherwise, we stop and deliver $\tilde{x}^k$ as y^k.

C. Experiment Parameter Setup

1) Size of P_L (N) and Weight Vectors (λ) in MOMAD Their settings are determined by a parameter H. More precisely, $\lambda^1, \ldots, \lambda^N$ constitute all the weight vectors in which each individual weight component takes a value from

$$\left\{\frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H}\right\}.$$

Therefore, the size of P_L (i.e., the number of weight vectors) is

$$N = C_{H+m-1}^{m-1}$$

where m is the number of objectives. Following the practice in [22] N is set as shown in Table VI.
2) *Max* is set to be 1 in our experiments.
3) θ in $N(x)$: It is set to be 0.02.
4) α in Perturbation: It is set to be 0.05.
5) Stopping Condition The algorithm stops after 1,500 generations in the case of two objectives, and 100 generations in other test instances.
6) Number of Independent Runs The algorithm is run 20 times on each test instance.

D. Algorithms in Comparison

1) MOTGA [47]: This algorithm divides the whole PF into a few small parts, each of them is approximated by an independent stage. Each stage consists of a number of consecutive genetic processes, each of which is guided by a different Tchebycheff aggregation function. The experimental results of MOTGA used in our comparison were obtained from its authors.
2) MEMOTS [48]: It is a memetic algorithm. A dynamic hypergrid in the objective space is used to guide selection. Tabu search is applied to improve each offspring. The experimental results of MEMOTS used in our comparison are from its authors' web[2].
3) 2PPLS [46]: It adopted a greedy heuristic to obtain an initial population. In the second phase (i.e., Pareto local search phase), the neighborhood is defined by using two candidate lists. The experimental results of 2PPLS used in our comparison are from its author's web. As pointed out in [46], 2PPLS could involve very heavy computational overhead if the number of objectives is more than two, there is no experimental results available on instances with three or more objectives[2].
4) MOEA/D-ACO [45]: In this experiment, the initial population was obtained by solving a set of weighted sum subproblems and the number of subproblems was the same as the number of ants. The CPU time used by this algorithm is the same as that by MOMAD, and the other parameters are set to the same values as in [45][3].

E. Comparison Results

The hypervolume values of the final solution sets produced by all the algorithms are plotted in Figs. 5–7. In computing the hypervolme, $y_i^*(i = 1, \ldots, m)$ is set to be 0. The mean C-metric values between MOMAD and the other algorithms on each instance are given in Table VII. The average running time used by each algorithm is given in Table VIII. We have the following observations.

1) In terms of both the hypervolume and C-metric values, MOMAD is much better than MOTGA and MEMOTS. For example, on the 500-3 instance, a solution set obtained by MOMAD can, on an average, dominate 79.8% of a solution set by MOTGA, and 1.0% vice versa; MOMEAD dominates MEMOTS by 79.3%, and 3.7% vice versa. On the same instance, as shown in Fig. 6, MOMAD has produced much larger hypervolume values than MOTGA and MEMOTS, which confirms the advantage of MOMAD.
2) On the three biobjective instances, MOMAD performs slightly better than or about the same as 2PPLS with respect to both the hypervolume and C-metric values. MOMAD provides a mechanism to combine fine and coarse search techniques. With the help of perturbation, MOMAD may flip away from local optima and therefore it is possible to find better solutions than 2PPLS.
3) Compared with MOEA/D-ACO, MOMAD can provide much better results on the nine instances in terms of both hypervolume and C-metric.
4) The CPU running time consumed by MOMAD is at the same magnitude as that by the other algorithms on these instances.

V. CONCLUSION AND FUTURE RESEARCH ISSUES

Evolutionary algorithms, decomposition approaches and Pareto local search methods are often used as basic elements in

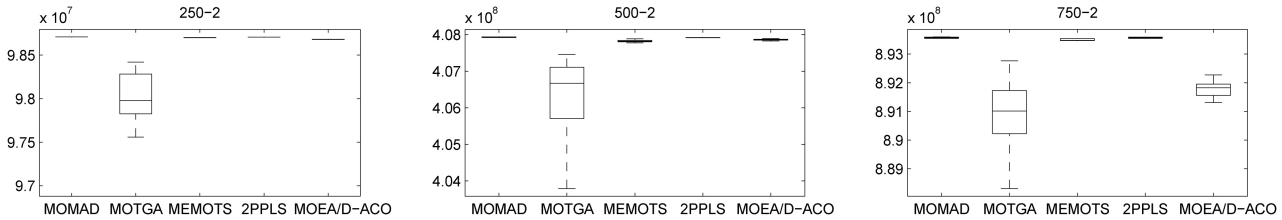

Fig. 5. From left to right: hypervolume values obtained by MOMAD, MOTGA, MEMOTS, 2PPLS, and MOEA/D-ACO on three mKP instances with two objectives.

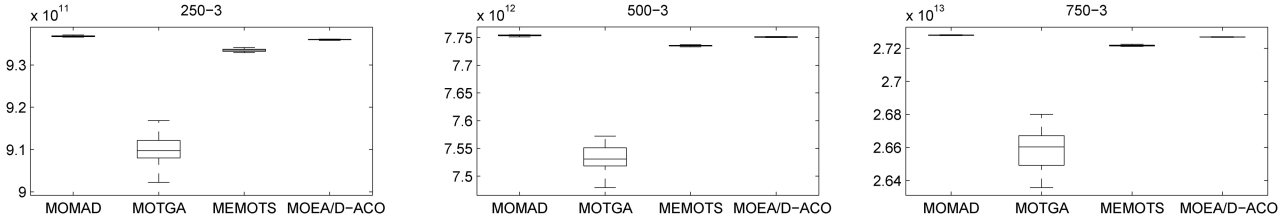

Fig. 6. From left to right: hypervolume values obtained by MOMAD, MOTGA, MEMOTS, and MOEA/D-ACO on three mKP instances with three objectives.

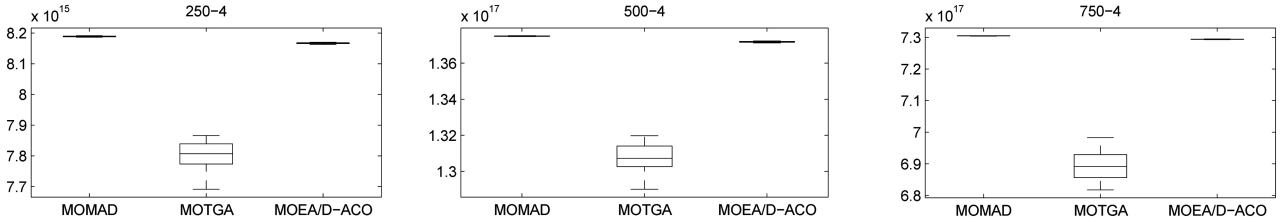

Fig. 7. Hypervolume values obtained by MOMAD, MOTGA, and MOEA/D-ACO on three mKP instances with four objectives (from left to right).

TABLE VII

COVERAGE VALUES (%) BETWEEN MOMAD AND MOTGA, MEMOTS, 2PPLS, MOEA/D-ACO FOR THE MKP

Instance	MOTGA		MEMOTS		2PPLS		MOEA/D-ACO	
	$C(A,B)$	$C(B,A)$	$C(A,B)$	$C(B,A)$	$C(A,B)$	$C(B,A)$	$C(A,B)$	$C(B,A)$
250-2	91.9	0.1	56.8	5.3	18.7	11.7	65.1	2.3
500-2	95.1	0.2	82.2	3.7	26.6	25.8	77.5	2.3
750-2	95.0	0.7	79.9	11.2	59.2	22.2	98.5	0.1
250-3	83.5	0.7	70.0	5.9	N.A.	N.A.	40.8	16.3
500-3	79.8	1.0	79.3	3.7	N.A.	N.A.	34.2	22.1
750-3	76.8	0.7	78.3	2.1	N.A.	N.A.	32.7	18.0
250-4	63.4	0.1	N.A.	N.A.	N.A.	N.A.	27.2	5.5
500-4	48.3	0.1	N.A.	N.A.	N.A.	N.A.	15.2	3.3
750-4	43.3	0.1	N.A.	N.A.	N.A.	N.A.	14.6	3.3

A corresponds to MOMAD.
B corresponds to the compared algorithm.
N.A. means unavailable.

TABLE VIII

RUNNING TIME (IN SECONDS) CONSUMED BY MOMAD, MOTGA, MEMOTS, 2PPLS, AND MOEA/D-ACO FOR THE MKP

Instance	MOMAD	MOTGA	MEMOTS	2PPLS	MOEA/D-ACO
250-2	7.4	1.6	5.0	5.0	7.4
500-2	24.5	7.7	23.0	26.0	24.5
750-2	45.2	20.8	59.0	51.0	45.2
250-3	1.9	2.9	11.0	N.A.	1.9
500-3	13.1	13.7	38.0	N.A.	13.1
750-3	19.9	35.6	95.0	N.A.	19.9
250-4	6.9	4.5	N.A.	N.A.	6.9
500-4	15.4	19.4	N.A.	N.A.	15.4
750-4	35.0	55.4	N.A.	N.A.	35.0

MOTGA, MEMOTS and 2PPLS were tested on a PC with Pentium 3.2GHz, 3.0GHz, and 3.0GHz CPU respectively. N.A. means that the result is not available.

designing hybrid heuristics for combinatorial MOPs. However, very little, if any, effort has been made to study how to combine these three elements together before. This paper has suggested a simple yet efficient multiobjective memtic algorithm, called MOMAD. It hybridizes all these three elements in one algorithmic framework. MOMAD provides a generic hybrid algorithmic framework to use problem specific knowledge and employ well developed single objective local search and heuristics, and Pareto local search methods for dealing with combinatorial multiobjective problems. It is a population based iterative method and thus an anytime algorithm. Extensive experiments have been conducted in this paper to study MOMAD and compare it with some other state-of-the-art algorithms on the multiobjective traveling salesman problem and the multiobjective knapsack problem. The experimental results show that our proposed algorithm outperforms or performs similarly to the best so far heuristics on these two problems.

The future research avenues include the following.

1) The study of MOMAD on other hard or real-world multiobjective problems. Such a study will be useful and necessary for understanding MOMAD and establishing some working principles for designing hybrid multiobjective algorithms.
2) The study of the combination of MOMAD with the DM's preference information [49]. It is very interesting to study how to use such information to guide problem decomposition and Pareto local search in the MOMAD framework so that the effort is concentrated in the PF parts which are of interest to decision makers.
3) The investigation of on-line dynamic allocation of computational resources to different PF parts in the MOMAD framework. It should be an effective way for improving the performance of MOMAD
4) The combination of MOMAD with machine learning techniques. It is worthwhile studying how to use machine learning techniques to model the MOP landscape and use it in the MOMAD framework.

The C++ source code of MOMAD can be downloaded from: *http://dces.essex.ac.uk/staff/zhang/*.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers and the Associate Editor.

REFERENCES

[1] K. Miettinen, *Nonlinear Multiobjective Optimization*. Norwell, MA, USA: Kluwer Academic, 1999.
[2] M. Ehrgott, *Multicriteria Optimization*. Berlin, Germany: Springer, 2005.
[3] Y. C. Liang and M. H. Lo, "Multi-objective redundancy allocation optimization using a variable neighborhood search algorithm," *J. Heurist.*, vol. 16, no. 3, pp. 511–535, 2010.
[4] E. Ulungu, J. Teghem, P. Fortemps, and D. Tuyttens, "MOSA method: A tool for solving multiobjective combinatorial optimization problems," *J. Multi-Criteria Decision Anal.*, vol. 8, no. 4, pp. 221–236, 1999.
[5] L. Paquete and T. Stützle, "Design and analysis of stochastic local search for the multiobjective traveling salesman problem," *Comput. Oper. Res.*, vol. 36, no. 9, pp. 2619–2631, 2009.
[6] A. Alsheddy and E. Tsang, "Guided Pareto local search based frameworks for biobjective optimization," in *Proc. IEEE CEC*, 2010, pp. 1–8.
[7] S. Saha, U. Maulik, and K. Deb, "A simulated annealing-based multiobjective optimization algorithm: AMOSA," *IEEE Trans. Evol. Comput.*, vol. 12, no. 3, pp. 269–283, Jun. 2008.
[8] C. García-Martínez, O. Cordón, and F. Herrera, "A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP," *Eur. J. Oper. Res.*, vol. 127, no. 1, pp. 116–148, 2007.
[9] R. Battiti and P. Campigotto, "Reactive search optimization: Learning while optimizing. an experiment in interactive multi-objective optimization," in *Proc. VIII MIC*, 2009, pp. 347–360.
[10] L. Paquete and T. Stützle, "A two-phase local search for the biobjective traveling salesman problem," in *Proc. 2nd Int. Conf. EMO*, 2003, pp. 479–493.
[11] T. Lust and J. Teghem, "Two-phase Pareto local search for the biobjective traveling salesman problem," *J. Heurist.*, vol. 16, no. 3, pp. 475–510, 2010.
[12] T. Lust and A. Jaszkiewicz, "Speed-up techniques for solving large-scale biobjective TSP," *Comput. Oper. Res.*, vol. 37, no. 3, pp. 521–533, 2010.
[13] E. Angel, E. Bampis, and L. Gourvès, "A dynasearch neighborhood for the bicriteria traveling salesman problem," in *Metaheuristics for Multiobjective Optimisation (LNEMS)*, vol. 535, X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, Eds. Berlin, Germany: Springer, 2004, pp. 153–176.
[14] L. Paquete, M. Chiarandini, and T. Stützle, "Pareto local optimum sets in the biobjective traveling salesman problem: An experimental study," in *Metaheuristics for Multiobjective Optimisation (LNEMS)*, vol. 535, X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, Eds. Berlin, Germany: Springer, 2004, pp. 177–200.
[15] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.
[16] J. Knowles and D. Corne, "The Pareto archived evolution strategy: A new baseline algorithm for Pareto multiobjective optimisation," in *Proc. IEEE CEC*, 1999, pp. 98–105.
[17] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.
[18] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 204–223, Apr. 2003.
[19] E. J. Hughes, "Multiple single objective Pareto sampling," in *Proc. IEEE CEC*, 2003, pp. 2678–2684.
[20] Y. Jin, T. Okabe, and B. Sendho, "Adapting weighted aggregation for multiobjective evolution strategies," in *Proc. 1st Int. Conf. EMO*, 2001, pp. 96–110.
[21] A. Jaszkiewicz, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem—A comparative experiment," *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 402–412, Aug. 2002.
[22] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.
[23] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Proc. 8th Int. Conf. PPSN*, 2004, pp. 832–842.
[24] Y. Mei, K. Tang, and X. Yao, "Decomposition-based memetic algorithm for multi-objective capacitated arc routing problem," *IEEE Trans. Evol. Comput.*, vol. 15, no. 2, pp. 151–165, Apr. 2011.
[25] V. A. Shim, K. C. Tan, and C. Y. Cheong, "A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 5, pp. 682–691, Sep. 2012.
[26] K. Sindhya, K. Miettinen, and K. Deb, "A hybrid framework for evolutionary multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 17, no. 4, pp. 495–511, Aug. 2013.
[27] S. Z. Martínez and C. A. C. Coello, "A direct local search mechanism for decomposition-based multi-objective evolutionary algorithms," in *Proc. IEEE CEC*, 2012, pp. 1–8.
[28] A. Waldock and D. Corne, "Multiple objective optimisation applied to route planning," in *Proc. 13th Annu. Conf. GECCO*, 2011, pp. 1827–1834.
[29] M. Asafuddoula, T. Ray, R. Sarker, and K. Alam, "An adaptive constraint handling approach embedded MOEA/D," in *Proc. IEEE CEC*, 2012, pp. 1–8.

[30] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, "Simultaneous use of different scalarizing functions in MOEA/D," in *Proc. 12th Ann. Conf. GECCO*, 2010, pp. 519–526.
[31] S. Pal, S. Das, A. Basak, and P. Suganthan, "Synthesis of difference patterns for monopulse antennas with optimal combination of array-size and number of subarrays—A multi-objective optimization approach," *Progr. Electromagnet. Res. B*, vol. 21, pp. 257–280, 2010.
[32] A. J. Nebro and J. J. Durillo, "A study of the parallelization of the multi-objective metaheuristic MOEA/D," in *Proc. LION*, 2010, pp. 303–317.
[33] Y. H. Chan, T. C. Chiang, and L. C. Fu, "A two-phase evolutionary algorithm for multiobjective mining of classification rules," in *Proc. IEEE CEC*, 2010, pp. 1–7.
[34] C. M. Chen, Y. P. Chen, T. C. Shen, and J. K. Zao, "Optimizing degree distributions in LT codes by using the multiobjective evolutionary algorithm based on decomposition," in *Proc. IEEE CEC*, 2010, pp. 1–8.
[35] N. Al Moubayed, A. Petrovski, and J. McCall, "A novel smart multi-objective particle swarm optimisation using decomposition," in *Proc. 11th Int. Conf. PPSN*, 2010, pp. 1–10.
[36] S. Z. Zhao, P. N. Suganthan, and Q. Zhang, "Decomposition-based multiobjective evolutionary algorithm with an ensemble of neighborhood sizes," *IEEE Trans. Evol. Comput.*, vol. 3, no. 16, pp. 442–446, Jun. 2012.
[37] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 28, no. 3, pp. 392–403, Aug. 1998.
[38] R. Battiti, M. Brunato, and F. Mascia, *Reactive Search and Intelligent Optimization*. Berlin, Germany: Springer, 2009.
[39] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 284–302, Apr. 2009.
[40] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with Gaussian process model," *IEEE Trans. Evol. Comput.*, vol. 12, no. 5, pp. 456–474, Jun. 2010.
[41] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model based multiobjective estimation of distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 41–63, Feb. 2008.
[42] D. Applegate, W. Cook, and A. Rohe, "Chained Lin-Kernighan for large traveling salesman problems," *INFORMS J. Comput.*, vol. 15, no. 1, pp. 82–92, 2003.
[43] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Oper. Res.*, vol. 21, no. 2, pp. 498–516, 1973.
[44] Y. P. Aneja and K. Nair, "Bicriteria transportation problem," *Manage. Sci.*, vol. 25, no. 1, pp. 73–78, 1979.
[45] L. Ke, Q. Zhang, and R. Battiti, "MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and ant colony," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1845–1859, Dec. 2013.
[46] T. Lust and J. Teghem, "The multi-objective multidimensional knapsack problem: A survey and a new approach," *Int. Trans. Oper. Res.*, vol. 19, no. 4, pp. 495–520, 2012.
[47] M. J. Alves and M. Almeida, "MOTGA: A multiobjective Tchebycheff based genetic algorithm for the multidimensional knapsack problem," *Comput. Oper. Res.*, vol. 34, no. 11, pp. 3458–3470, 2007.
[48] T. Lust and J. Teghem, "MEMOTS: A memetic algorithm integrating tabu search for combinatorial multiobjective optimization," *RAIRO Oper. Res.*, vol. 42, no. 1, pp. 3–33, 2008.
[49] R. Battiti and A. Passerini, "Brain-computer evolutionary multi-objective optimization (BC-EMO): A genetic algorithm adapting to the decision maker," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 671–687, Oct. 2010.

Liangjun Ke received the B.Sc. and M.Sc. degrees in mathematics from Wuhan University, Wuhan, China, in 1998 and 2001, respectively, and the Ph.D. degree in systems engineering from Xi'an Jiaotong University, Shaanxi, China, in 2008.

He is currently an Associate Professor with Xi'an Jiaotong University, China. From 2011 to 2012, he visited the School of Computer Science and Electronic Engineering, University of Essex, Colchester, U.K. His current research interests include optimization theory and applications, especially, multiobjective optimization, evolutionary computation and robust optimization.

Qingfu Zhang (M'01–SM'06) received the B.Sc. in mathematics from Shanxi University, Shanxi, China, in 1984, the M.Sc. in applied mathematics, and the Ph.D. in information engineering from Xidian University, Shanxi, in 1991 and 1994, respectively.

He is currently a Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong, a Professor on leave from the School of Computer Science and Electronic Engineering, University of Essex, Colchester, U.K., and a Changjiang visiting chair Professor with Xidian University. He holds two patents and is the author of many research publications. His current research interests include evolutionary computation, optimization, neural networks, data analysis, and their applications.

Dr. Zhang is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS. He is also an Editorial Board Member of four other international journals. MOEA/D, a multiobjective optimization algorithm developed in his group, won the Unconstrained Multiobjective Optimization Algorithm Competition at the Congress of Evolutionary Computation 2009, and was awarded the 2010 IEEE Transactions on Evolutionary Computation Outstanding Paper Award.

Roberto Battiti (F'09) received the Laurea degree in Physics from the University of Trento, Italy, in 1985 and the Ph.D. in Computation and Neural Systems from the California Institute of Technology (Caltech), USA, in 1990.

He is now Full Professor of computer science at Trento University, deputy director of the DISI Department (Electrical Engineering and Computer Science) and delegate for technology transfer. His main research interests are heuristic algorithms for problem-solving, "LION" techniques combining machine learning and intelligent optimization, in particular Reactive Search Optimization (RSO), which aims at embodying solvers with internal machine learning techniques, data mining, big data and visualization.

R. Battiti is a fellow of the IEEE, author of higly cited publications, and founder of the startups Reactive Search SrL and Lionsolver Inc. Full details about interest and research activities can be found on the web at http://lion.disi.unitn.it/∼battiti/.