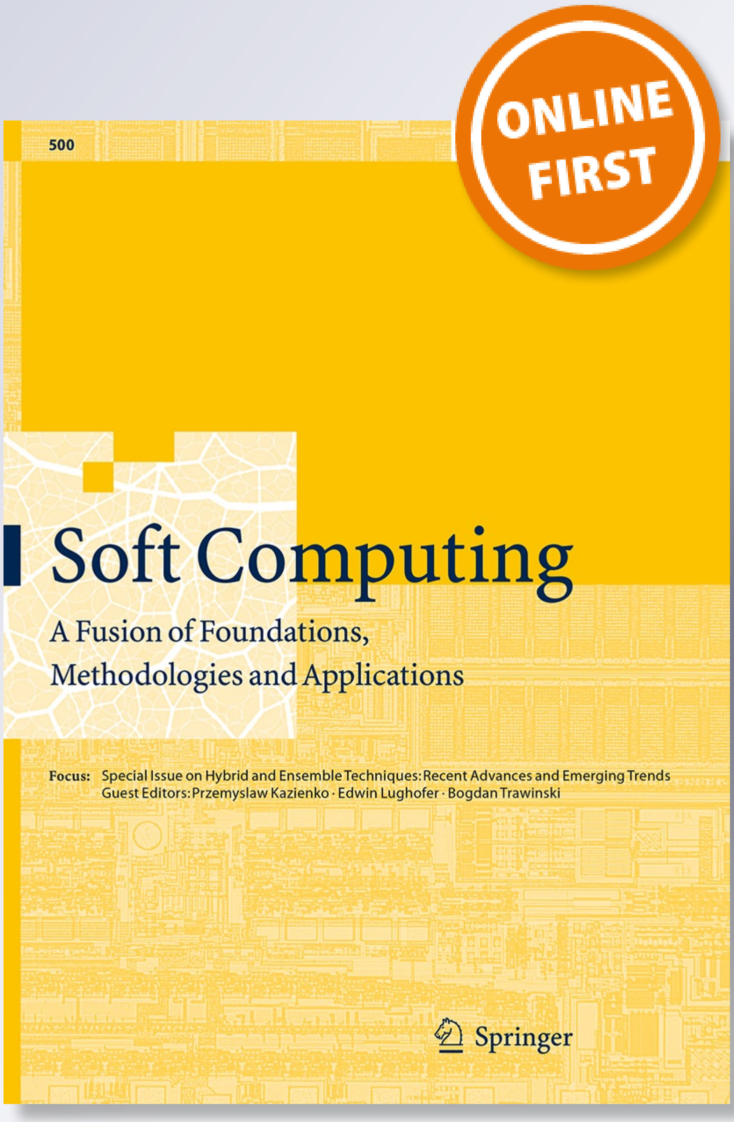
An adaptive memetic framework for multi-objective combinatorial optimization problems: studies on software next release and travelling salesman problems

Xinye Cai, Xin Cheng, Zhun Fan, Erik Goodman & Lisong Wang

Soft Computing
A Fusion of Foundations, Methodologies and Applications

ISSN 1432-7643

Soft Comput
DOI 10.1007/s00500-015-1921-0


ONLINE FIRST
500
Soft Computing
A Fusion of Foundations, Methodologies and Applications
Focus: Special Issue on Hybrid and Ensemble Techniques: Recent Advances and Emerging Trends
Guest Editors: Przemyslaw Kazienko · Edwin Lughofer · Bogdan Trawinski
Springer

Soft Comput
DOI 10.1007/s00500-015-1921-0

METHODOLOGIES AND APPLICATION

Series A

An adaptive memetic framework for multi-objective combinatorial optimization problems: studies on software next release and travelling salesman problems

Xinye Cai[1,2] · Xin Cheng[2] · Zhun Fan[3] · Erik Goodman[4] · Lisong Wang[2]

Abstract In this paper, we propose two multi-objective memetic algorithms (MOMAs) using two different adaptive mechanisms to address combinatorial optimization problems (COPs). One mechanism adaptively selects solutions for local search based on the solutions' convergence toward the Pareto front. The second adaptive mechanism uses the convergence and diversity information of an external set (dominance archive), to guide the selection of promising solutions for local search. In addition, simulated annealing is integrated in this framework as the local refinement process. The multi-objective memetic algorithms with the two adaptive schemes (called uMOMA-SA and aMOMA-SA) are tested on two COPs and compared with some well-known multi-objective evolutionary algorithms. Experimental results suggest that uMOMA-SA and aMOMA-SA outperform the other algorithms with which they are compared. The effects of the two adaptive mechanisms are also investigated in the paper. In addition, uMOMA-SA and aMOMA-SA are compared with three single-objective and three multi-objective optimization approaches on software next release problems using real instances mined from bug repositories (Xuan et al. IEEE Trans Softw Eng 38(5):1195–1212, 2012). The results show that these multi-objective optimization approaches perform better than these single-objective ones, in general, and that aMOMA-SA has the best performance among all the approaches compared.

Keywords Multi-objective combinatorial optimization · Memetic algorithms · Decomposition-based method · Local search · Adaptation

Communicated by V. Loia.

✉ Zhun Fan
zfan@stu.edu.cn

Xinye Cai
xinye@nuaa.edu.cn

Xin Cheng
chengxin0223@126.com

Erik Goodman
goodman@egr.msu.edu

Lisong Wang
wangls@nuaa.edu.cn

1 The 28th Research Institute of China Electronic Technology Group Corporation, Nanjing, People's Republic of China

2 College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu 210016, People's Republic of China

3 Guangdong Provincial Key Laboratory of Digital Signal and Image Processing Techniques, Department of Electronic Engineering, School of Engineering, Shantou University, Shantou, Guangdong, People's Republic of China

4 BEACON Center for the Study of Evolution in Action, Michigan State University, East Lansing, MI, USA

1 Introduction

Combinatorial optimization problems (COPs), such as the travelling salesman problem, knapsack problem, next release problem, flowshop scheduling problem and vehicle routing problem (Arroyo and Armentano 2005; Durillo et al. 2011; Papadimitriou and Steiglitz 1998; Peng et al. 2009; Sato et al. 2007; Tan et al. 2006), have been extensively studied for many years because of their widespread application in real life. Most COPs are NP-hard, which means that no exact algorithms are known to solve these problems within polynomial computation time, due to their huge search spaces, many local optima and complex constraints. Numerous previous publications have shown that meta-heuristics, such as evolu-

Published online: 21 December 2015

tionary algorithms (EAs), are very suitable to address such problems (Borges and Hansen 1998). As a stochastic search technique, EAs have become prevalent for a wide range of engineering optimization problems. Originally, EAs mimic the evolutionary process to sample, learn and adapt from a population of solutions, by applying genetic operations such as crossover and mutation.

In general, evolutionary algorithms are capable of exploring and exploiting promising regions of the search space. However, it takes them a relatively long time to locate the exact local optimum within the region of convergence when facing complex optimization problems (Grosan and Abraham 2007). Memetic algorithms (MAs) (Chen et al. 2011; Nguyen et al. 2009; Ong et al. 2010) are usually considered as extensions of EAs, which introduce the individual learning as a separate local search process for speeding up the search process. In MAs, EAs usually explore the entire search space as a global optimizer for locating the most promising regions with a population of individuals, while heuristic local searches exploit the located regions by improving the individuals in the population.

Real-world applications of combinatorial optimization usually require several conflicting objectives to be optimized at once. Consequently, memetic algorithms have been extended to multi-objective forms, which are usually called multi-objective memetic algorithms (MOMAs), to address complex multi-objective COPs. Very naturally, many local search techniques in single-objective heuristics, such as iterative local search (Paquete and Sttzle 2009), guided local search (Alsheddy and Tsang 2010), tabu search (Ulungu et al. 1999), variable neighborhood search (Liang and Lo 2010), ant colony optimization (García-Martínez et al. 2007) and simulated annealing (Maulik et al. 2008) have been generalized to solve multi-objective COPs.

In multi-objective optimization problems, the improvements of one objective usually lead to the degradation for at least one other objective. Thus, different from a single-objective optimization problem where one optimal solution exists, a set of non-dominated solutions co-exists for a multi-objective optimization problem. These solutions represent the trade-offs among the multiple objectives. In this situation, how to assign fitness values to a solution becomes very critical to allow selecting fitter and more diverse solutions in MOMAs. In Ke et al. (2014), Sindhya et al. (2013), Pareto local search (dominance) and aggregation (decomposition) are two different fitness assignment schemes for local search, which have been frequently used in MOMAs.

- *Pareto local search* Lust and Jaszkiewicz (2010), Lust and Teghem (2010) explores the neighborhood regions of a solution to find more diverse non-dominated solutions for updating the current population. The non-dominated solutions are selected based on Pareto dominance and they are further selected by some diversity maintenance or crowding scheme.
- *Aggregation method* Ke et al. (2014), Shim et al. (2012) transforms a multi-objective optimization problem into a number of single-objective subproblems and solve them simultaneously. The objective function in each subproblem can be a linear or nonlinear weighted aggregation function of all the objective functions in question. Under mild conditions, an optimal solution to one single-objective subproblem is one of many Pareto-optimal solutions for the multi-objective optimization problem. Consequently, a single-objective local search method can be directly applied to multiple single-objective subproblems for locating a set of Pareto optimal solutions.

For example, Ishibuchi and Murata (1996) proposed a multi-objective genetic local search based on aggregation for a flow shop scheduling problem. Jaszkiewicz (2002) designed a weighted-sum-function-based MOMA for a knapsack problem. Shim et al. (2012) proposed a MOMA based on aggregation to tackle multi-objective travelling salesman problems. In Shim et al. (2012), an estimation of distribution algorithm is adopted as the global search engine and combined with three different local search methods based on meta-heuristics. Different from Shim et al. (2012) that concentrates on local search based on a weighted sum, the memetic-based Pareto archived evolution strategy (M-PAES), presented by Knowles and Corne (2000), uses Pareto-ranking-based selection for local search. Later, Lust and Jaszkiewicz (2010) proposed a two-phase Pareto local search (2PPLS). The 2PPLS combines Pareto local search and aggregation using two phases. Phase one generates promising solutions for local search using a number of linear aggregation formulations. Phase two applies a Pareto local search to every solution generated in phase one to find Pareto optimal solutions. Very recent work in Ke et al. (2014), Shim et al. (2014) integrates Pareto local search into a popular multi-objective evolutionary algorithm based on decomposition (MOEA/D), which shows that the combination of aggregation and Pareto local search is very promising to further improve the performance of MOMAs.

Motivated by Ke et al. (2014), Lust and Jaszkiewicz (2010), Shim et al. (2014), this paper is dedicated to proposing an adaptive multi-objective memetic framework based on both Pareto local search and aggregation. Over the recent decades, adaptation has been considered as one of the most promising areas of research in memetic algorithms and has attracted much attention. For many real-world applications where limited insight has been provided, it is difficult to design a specific MA without adaptation that can still work well. Adaptive MAs take advantage of the information about the given problem and adapt themselves to the problem's characteristics during the search process, for accelerating the search process and achieving better performance. For instance, Ong and Keane (2004) proposed a type of Meta-Lamarckian learning in which refinement procedures are cooperative and competitive according to adaptive strategies. Krasnogor and Gustafson (2004) presented a self-generating mechanism by which various local search mechanisms to be used in a memetic algorithm are executed adaptively. Nguyen et al. (2007) investigated the impacts of the refinement frequency, selection of individual subset and intensity of refinement on MAs through empirical experiments. Later, they proposed a probabilistic framework to discuss the adaptation of single-objective MAs (Nguyen et al. 2009). Ishibuchi et al. (2003) investigated the impact of frequency of refinement in both single- and multi-objective contexts for permutation flowshop scheduling, where only the elite individuals of the population undergo refinement.

In a more recent survey Chen et al. (2011), summarized adaptation schemes into the following categories: (1) frequency of refinement and selection of the individual subset to undergo refinement, (2) intensity of refinement and (3) choice of procedures to conduct refinement. Among them, frequency of refinement and individual subset selection defines the proportion of a population that should undergo the refinement process (Sudholt 2006), which aims to balance the amount of computational budget allocated for global and local search. It has been proved in Krasnogor and Smith (2005), Nguyen et al. (2007) and Nguyen and Ong (2009) that adaptive selection of the individual subset to undergo refinement helps enhance overall search productivity. Intensity of refinement defines the amount of computational budget allocated to a local refinement process. Relevant studies have confirmed its significant impact on performance of memetic algorithms (Droste et al. 2002). The adaptive selection of refinement procedures aims to address the issue of what refinement procedure, among multiple refinement methods, should be applied during the various stages of the search process. It has demonstrated its significance to memetic algorithms on a variety of problems (Ong et al. 2006).

However, very little research has been conducted on the adaptation of MOMAs. Among the above adaptation issues, the adaptive selection of refinement procedures was investigated based on cross-dominance (Caponio and Neri 2009) and resource productivity criteria (Bosman 2012). To the best of our knowledge, frequency of refinement and individual subset selection and Intensity of refinement have been little investigated in MOMAs. In this paper, we adopt two adaptive mechanisms to address the issue of frequency of refinement and individual subset selection, in our multi-objective memetic framework, to further enhance its search ability with respect to convergence and diversity.

The main contributions of this paper are as follows:

– A multi-objective memetic framework is proposed with simulated annealing integrated as the local search metaheuristic, to tackle multi-objective COPs.
– We investigated the impacts of refinement frequency and selection of individual subset in our multi-objective memetic framework. Furthermore, we adopt and investigate two adaptive mechanisms, namely utility-based and external-set-guided mechanisms, to select promising individual subsets for local search.
– Our algorithm was validated on the multi-objective software next release problem (MONPR) with both synthetic and real test instances mined from bug repositories (Xuan et al. 2012) and on the multi-objective travelling salesman problem (MOTSP). For the real test instances of MONRP, our MOMAs were compared with the state-of-the-art single-objective approaches as well as well-known MOEAs. The results showed that our MOMAs outperformed these other approaches.

The rest of this paper is organized as follows: Since this paper focuses on multi-objective optimization problems, Sect. 2 revisits basic concepts of multi-objective optimization problems (MOPs). The introduction of MOEAs is also included in this section. Section 3 analyses some design issues in MOMAs. Section 4 is mainly dedicated to the detailed descriptions of the multi-objective memetic framework and two adaptive mechanisms for local search. Section 5 introduces two representative benchmark multi-objective COPs. Experimental studies and discussions are detailed in Sect. 6. The effects of the local search process as well as the two adaptive mechanisms are also investigated in this section. The final conclusions of this paper are provided in Sect. 7.

2 Background

Since this paper focuses on multi-objective optimization, this section introduces multi-objective optimization problems and multi-objective evolutionary algorithms as follows.

2.1 A Multi-objective optimization problem

A multi-objective optimization problem (MOP) can be stated as follows:

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

where Ω is the decision space and $F : \Omega \to R^m$ consists of m real-valued objective functions. The attainable objective set is $\{F(x)|x \in \Omega\}$. In the case when Ω is a finite set, (1) is called a discrete MOP.

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 non-dominated in S if no other point in S can dominate it. A point $x^* \in \Omega$ is Pareto-optimal if $F(x^*)$ is non-dominated 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 deterioration in at least one other objective. The set of all Pareto-optimal points is called the Pareto set (PS) and the set of all Pareto-optimal objective vectors is called the Pareto front (PF) (Miettinen 1999).

2.2 Multi-objective evolutionary algorithms

In order to effectively tackle MOPs, a suitable framework that assigns fitness and maintains the diverse trade-off Pareto optimal solutions must be applied. Based on the use of different schemes for fitness assignment, MOEAs can be further divided into three categories: dominance-based (Deb 2001; Deb et al. 2002; Zitzler et al. 2002), indicator-based and decomposition-based (Zhang and Li 2007) MOEAs.

Well-known representatives of dominance-based MOEAs include SPEA2 (Zitzler et al. 2002) and NSGA-II (Deb 2001; Deb et al. 2002). In these algorithms, diversity is maintained by eliminating solutions with high density (e.g., crowding distance methods in NSGA-II and nearest neighbours method in SPEA2).

The multi-objective evolutionary algorithm based on decomposition is a popular decomposition-based MOEA (Zhang and Li 2007). It decomposes a MOP into several single-objective subproblems, which are defined by a weighted aggregation function of all the objective functions in question, and all the subproblems are optimized in a collaborative manner. One solution is associated with each of the subproblems. Two subproblems are called neighbours if their weight vectors are close to each other. MOEA/D explores correlation relationships among neighbouring subproblems to speed up its search. In MOEA/D, the replacement of solutions (population convergence) is determined by their aggregation function values, and population diversity is achieved by distributing the subproblems widely through weights. MOEA/D and its variants have been tested and proven very effective on various benchmark functions and real-world problems in the literature (Chang et al. 2008; Ishibuchi et al. 2009; Kafafy et al. 2012; Li and Zhang 2009; Konstantinidis et al. 2010; Peng and Zhang 2012; Peng et al. 2009). The goal of this paper was to design a multi-objective memetic framework that is able to combine the advanced characteristics of MOEA/D and Pareto local search.

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

3 Two important design issues in MOMAs

We address mainly the following two important design issues of MOMAs, as follows.

3.1 Fitness assignment scheme for local search

According to Ke et al. (2014), Sindhya et al. (2013), both Pareto local search (dominance) and aggregation (decomposition) are used as fitness assignment schemes for local search in the framework of MOMAs. In this paper, we adopt both fitness assignment schemes through two populations in MOMA: a decomposition population and an external set. The external set is used as the output of the algorithm—the final solution set for the MOP. In addition, the information extracted from the external set can be used to guide the search direction in the decomposition population.

3.2 Adaptation

As surveyed in Sect. 1, very little research has been conducted on the adaptation of MOMAs. However, existing research on adaptation in the context of multiobjective optimization can be inspirations for designing MOMAs. For multiobjective optimization, the algorithm is desirable to balance between convergence and diversity for obtaining good approximation to the set of Pareto optimal solutions. Convergence can be defined as the distance of solutions towards the PF, which should be as small as possible. Diversity can be defined as the spread of solutions along the PF, which should be as uniform as possible. The use of convergence and diversity information helps to locate "promising regions" and guide more efficient search. For example, MOEA/D-DRA (Zhang et al. 2009), a state-of-the-art variant of MOEA/D adopted a dynamic mechanism of resource allocation to provide different computational efforts for different subproblems based on each subproblem's convergence information (utility). More recently, an adaptive mechanism, which extracts both the convergence and diversity information to guide the search, was proposed in Cai et al. (2014) and Li et al. (2014). The proposed algorithm resorts to an external archive to guide the search.

In this paper, we proposed two adaptive memetic algorithms to address combinatorial optimization problems. Two adaptive mechanisms, one based on utility (Zhang et al. 2009) and the other based on the external archive (Cai et al. 2014; Li et al. 2014), are adopted to determine the frequency of refinement and the selection of solutions for refinement during different phases of the search process, in our multiobjective memetic framework. For convenience, the adaptive memetic algorithm that uses utility is named uMOMA-SA and the one that adopts the external archive to guide the local search is named aMOMA-SA in this paper.

The external-set-guided adaptive mechanism is inspired by EAG-MOEA/D (Cai et al. 2014). So this section discusses the similarities and differences between aMOMA-SA and EAG-MOEA/D. Both aMOMA-SA and EAG-MOEA/D adopt adaptive selection of solutions. However, these two algorithms are different in the following aspects:

1. The frameworks of the aMOMA-SA and EAG-MOEA/D are different.
   (a) aMOMA-SA is essentially a multiobjective memetic algorithm, which uses external-set-guided selection for the local search (simulated annealing) while EAG-MOEA/D does not have any local search. It only uses external set to guide the global search.
   (b) In EAG-MOEA/D, all the generated solutions have been put in a temporary population L, which later is used to update the external set A. However, in aMOMA-SA, the update of population L is based on Boltzmann criterion (see Sect. 4.2.4).
2. The adaptive mechanisms for choosing subproblems are different. In EAG-MOEA/D, the probability for choosing a subproblem is calculated based on the contributions of each subproblem on the external set over the last certain number of generations. However, in aMOMA-SA, the probability is calculated based on the contributions of each subproblem on the current external set.

4 The proposed multi-objective memetic algorithms

Our multi-objective memetic algorithm with simulated annealing as its local search method (MOMA-SA) decomposes a MOP into N single-objective optimization subproblems by aggregating objective functions with different weight vectors. In principle, we can adopt any aggregation method for this purpose. For simplicity, we use the weighted sum approach with N weight vectors:

$$\lambda^j = (\lambda_1^j, \dots, \lambda_m^j)\, j = 1, \dots, N. \tag{2}$$

where $\lambda^j \in R_+^m$ and $\sum_{i=1}^m \lambda_i^j = 1$, m is the number of objectives. Subproblem k is:

$$\begin{aligned} &\text{minimize } g_k^{ws}(x) = \textstyle\sum_{i=1}^m \lambda_i^j f_i(x) \\ &\text{subject to } \quad x \in \Omega \end{aligned} \tag{3}$$

For each $k = 1, \dots, N$, set $B(k)$ stores the indices of the T closest weight vectors to λ^k in terms of the Euclidean distance. Subproblem i is defined as a neighbour of subproblem k, when $i \in B(k)$.

The following solution are maintained during the evolutionary process:

– Decomposition population set $P = \{x_1, \dots, x_k, \dots, x_N\}$, where x_k is the best solution associated with subproblem k.
– External set (dominance archive) A, which contains N solutions selected by fast-non-dominated-sorting and crowding-distance-assignment in NSGA-II (Deb 2001; Deb et al. 2002).
– L, which stores solutions after undergoing local search.
– Y, which stores solutions after undergoing global search.

The pseudocode of the framework is presented in Algorithm 1. It works as follows:

Step 1 **Initialization:** Initialize P and A.
Step 2 **Local Search:** Use a mechanism to select solutions from P (in Sect. 4.3) to undergo local search; the generated new solution set are stored in L.
Step 3 **Global Search:** Perform global search on P to generate a new solution set Y.
Step 4 **Update of Population and External Set:** Use L and Y to update P and A.
Step 5 **Stopping Criterion:** If preset criteria are satisfied, output A. Otherwise, go to Step 2.

More detailed descriptions of Steps 1–4 of Algorithm 1 are given as follows:

4.1 Initialization

A MOP is originally decomposed into N subproblems based on Eqs. 2 and 3. The decomposition population P is randomly generated and assigned to the external set A. The T closest neighbours, in terms of Euclidean distance, to each weight vector are determined. Therefore, subproblem i has T neighbours, denoted as $B(i) = \{i_1, \dots, i_T\}$. The whole process of initialization is shown in Step 1.

4.2 Local search

The procedures of local search are demonstrated in Step 2 as follows. In Step 2a, using one of the following mechanisms, N solutions are chosen from P for local search. This is actually to address the issue of the frequency of refinement and individual subset selection in Sudholt (2006). In this work, one baseline and two adaptive mechanisms for selecting individuals for local search are adopted as follows.

4.2.1 Global selection

When we equally select all the solutions of the population set P for local search, we are actually considering all the subproblems are equally promising, which is against our motivation of the adaptive mechanisms. However, it can be thought of a baseline mechanism; and the MOMA framework with this mechanism is named gMOMA-SA for comparison.

```
Algorithm 1: MOMA-SA
Input:
  1. A multi-objective COP;
  2. a stopping criterion;
  3. $N$: the population size of $P$ and $A$, as well as the number of subproblems;
  4. a uniform spread of $N$ weight vectors: $\boldsymbol{\lambda}^1, \ldots, \boldsymbol{\lambda}^N$;
  5. $T$: the size of the neighborhood of each subproblem;
Output: External set $A$;
Step 1: Initialization:
  a) Decompose a multi-objective COP into $N$ subproblems using $\boldsymbol{\lambda}^1, \ldots, \boldsymbol{\lambda}^N$.
  b) Randomly initialize a population $P = \{x_1, \ldots, x_N\}$.
  c) External set is initialized as $P$: $A = P$.
  d) Calculate the Euclidean distance between any two weight vectors and obtain $T$ closest weight vectors to each weight vector. For each $i = 1, \ldots, N$, set $B(i) = \{i_1, \ldots, i_T\}$, where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the $T$ closest weight vectors to $\lambda^i$.
Step 2: Local search
  a) Select $N$ solutions from $P$, where $Index = \{k_1, ..., k_N\}$ are the indices of the selected solutions in $P$.
  b) Conduct a local search on the selected solutions: Set $L = \emptyset$, $(P, L) = SA(P, Index, L)$.
Step 3: Global search
    For each $i \in P$, do:
  a) Randomly choose two indices $k$ and $l$ from $B(i)$.
  b) Apply genetic operators on $x_k$ and $x_l$ to generate $y_j \in Y$ for Subproblem $i$.
    End do
Step 4: Update of the population and external set
    /* Use $Y$ to update decomposition population $P$*/
    For each $i \in P$, do:
  a) Use $Y$ to update neighboring subproblems: For each index $k \in B(i)$, if $g^{ws}(y_i|\lambda^k) \leq g^{ws}(x_k|\lambda^k)$, then set $x_k = y_i$ and $F(x_k) = F(y_i)$.
    End do
    /* Use $L$ to update decomposition population $P$*/
    For each $j \in L$, do:
  b) Use $L$ to update neighboring solutions: If $L_j$ is generated from subproblem $i$, for each index $k \in B(i)$, if $g^{ws}(L_j|\lambda^k) \leq g^{ws}(x_k|\lambda^k)$, then set $x_k = L_j$ and $F(x_k) = F(L_j)$.
    End do
    /* Use $Y$ and $L$ to update external set $A$*/
  c) $Z = A \cup Y \cup L$; apply fast-non-dominated sorting and crowding-distance-assignment in NSGA-II[42] on $Z$ to obtain the best $N$ solutions; and store them as the new $A$.
Step 5: Termination
  a) If stopping criteria is satisfied, terminate the algorithm and output $A$. Otherwise, go to Step 2.
```

4.2.2 Utility-based selection

This solution selection mechanism for local search is inspired by MOEA/D-DRA (Zhang et al. 2009), as follows;

For subproblem (solution) i of the decomposition population P, the utility π^i is calculated as the progress of that subproblem and is updated every 50 generations, as shown in Eqs. 4 and 5.

$$\pi^i = \begin{cases} 1 & \text{if } \triangle^i > 0.001 \\ \left(0.95 + 0.05\frac{\triangle^i}{0.001}\right)\pi^i & \text{otherwise,} \end{cases} \tag{4}$$

where $\triangle^i$ is defined as the relative decrease in the objective value of subproblem i, considered as a minimization problem.

$$\triangle^i = \frac{\text{Old function value} - \text{New function value}}{\text{Old function value}} \tag{5}$$

In Eq. 4, if the progress ratio for subproblem i ($\triangle^i$ in Eq. 5) between the current generation and 50 generations earlier is larger than a preset threshold value (0.001), the utility value π^i of subproblem i remains at 1. Otherwise, π^i for subproblem i is reduced accordingly.

Then, a size-10 tournament selection is applied to choose solutions for local search based on the utility values given in Eq. 4. More specifically, the solution with the highest utility value π out of 10 randomly selected solutions from P is chosen for local search. This process is repeated N times until N solutions are chosen for local search. The MOMA framework with this mechanism is named uMOMA-SA in this paper.

4.2.3 External-set-guided selection

In our MOMA framework, the dominance archive has been updating in each generation and outputted as the final solution set. Undoubtedly, it contains convergence and diversity information during the whole optimization process. Such information may be very valuable for guiding the selection of solutions to undergo local search (Cai et al. 2014; Li et al. 2014).

If a solution survives in A after NSGA-II selection, it makes a contribution to A. Thus the number of such solutions generated through local search of a subproblem records the convergence and diversity information of this subproblem. In other words, we can use the number of "successful" solutions of each subproblem to estimate its potential.

Let us assume $ds_{i,g}$ is the number of successful solutions generated from subproblem i at generation g. The total number of successful solutions at generation g is used to compute the probability for each subproblem to be selected for local search, as follows.

At generation g, the probability of choosing subproblem i $(i = 1, 2, \dots, N)$ is

$$\text{prob}_{i,g} = \frac{D_{i,g}}{\sum_{i=1}^{N} D_{i,g}}, \tag{6}$$

where

$$D_{i,g} = \frac{ds_{i,g}}{N} + \epsilon, \quad (i = 1, 2, \dots, N); \tag{7}$$

$D_{i,g}$ is the proportion of solutions generated from subproblem i at generation g, $ds_{i,g}$ is the number of non-dominated solutions in A generated from subproblem i and the total number of non-dominated solutions in A is N. A small constant value $\epsilon = 0.002$ is adopted as a floor to avoid possible zero selection probabilities. We also normalize $D_{i,g}$, making the $D_{i,g}$ for all subproblems sum to 1.

Roulette wheel selection is applied to select a subproblem based on the probabilities calculated by Eq. 6. This process is repeated N times until N solutions have been selected for local search. For convenience, the MOMA framework with external-set-guided local search is named aMOMA-SA.

4.2.4 *Simulated annealing as local search*

In Step 2(b) of Algorithm 1, local search is applied on the selected solution set $\{x_{k_1}, \dots, x_{k_N}\}$, where Index $= \{k_1, \dots, k_N\}$ is the index of selected solutions in P. A new solution set L, which is used to store the solution set generated by local search, is set to $\emptyset$ before local search.

In general, any local search meta-heuristic can be integrated into our proposed multi-objective memetic framework. In this paper, simulated annealing (SA) (Rodriguez-Alvarez et al. 2004) is adopted for simplicity. The pseudocode of $(P, L) = \text{SA}(P, \text{Index}, L)$ is described in Algorithm 2.

The basic operation in SA is the random perturbation of a solution to search for a local neighbour. For each selected solution, a bit string of the solution is flipped to generate a new solution (binary case). The resulting solution is evaluated and compared with the original one. In the solution update step, SA accepts superior solutions and allows the acceptance of inferior solutions based on a Boltzmann criterion $\text{random}(0, 1) < \exp(F(y_j) - F(x_{k_i})/Te)$, where $\text{random}(0, 1)$ is a randomly generated floating number between 0 and 1; $F(y_j)$ and $F(x_{k_i})$ are weighted objective values of solutions with regard to subproblem k_i. Usually, Te is set to a relatively high value at the beginning; and it gradually decreases based on the annealing schedule factor α, where $\alpha \in (0,1)$. Based on the above criterion, inferior solutions have a relatively high probability to survive at the early stages, and this probability gradually decreases during the search process.

In addition, solution set L is used to store "good" solutions generated by local search in the following two conditions:

- A solution y_j that is better than the original one $(F(y_j) < F(x_{k_i}))$, with regard to subproblem k_i.
- A solution that is worse than the original one, but fits the Boltzmann criterion $(\text{random}(0, 1) < \exp(F(y_j) - F(x_{k_i})/Te))$

The former condition guarantees that a superior solution, in terms of a lower (better) weighted objective value in the associated subproblem, will be accepted in L. However, some slightly inferior solutions, according to the latter condition, will also be accepted based on the Boltzmann criterion. In addition to its normal motivation in SA, another reason for adopting the latter condition is motivated by the fact that a slightly inferior solution still may be a non-dominated solution in the external set (dominance archive) and so be worthy of storing.

An alternative method is to store all the solutions generated by local search in L. However, because J local searches are applied on each selected solution, a total of $J*N$ solutions are generated by local search. Such a large set is bound to increase the computational burden when sorting the merged set of A, L and P to obtain the new A in the process of updating A (Step 4d in Algorithm 1).

```
Algorithm 2 : $(P, L) = SA(P, Index, L)$
Input: $Index = \{k_1, \dots, k_N\}$, $P = \{x_1, \dots, x_N\}$, $L$ and temperature($Te$),
Output: $P$ and $L$
    For $i = 1$ to $N$(For each $k_i \in Index$)
    /* $N$ is number for solutions undergoing local search*/
      Initial solution: $x_{k_i}$
      For $j = 1$ to $J$
      /* $J$ is the number of local neighbors */
2:      Reproduction: Create local neighbor $(y_j)$ by flipping a
        single bit of $x_{k_i}$.
        Evaluation: Calculate the fitness value of $y_j$, $F(y_j)$
4:      Update solution:
        If $F(y_j) < F(x_{k_i})$ then
          $x_{k_i} = y^j$
6:        $L = y_j \cup L$
        else if random(0,1) < exp($F(y^j)$-$F(x_{k_i})$ / $Te$) then
          $x_{k_i}$=$y^j$
8:        $L = y^j \cup L$
        End if
      End for $j$
    End for $i$

    $Te = \alpha * Te$; 0< $\alpha$ < 1
```

4.3 Global search

The procedures of global search process are explained in Step 3. For each subproblem i, two parent solutions are randomly selected from its T neighbouring subproblems, as illustrated in Step 3a. Subsequently, genetic operators are applied to them for generating an offspring solution y_i, in the following Step 3b. For software next release problem, a one-point crossover and a bit-wise mutation operator are applied. For travelling salesman problem, a position based crossover and an exchange mutation operator (Larrañaga et al. 1999) are adopted. By repeating Step 3a–3b N times, an offspring population $Y = \{y_1, \dots, y_N\}$ can be obtained.

4.4 Update of the population and external set

The update of the population P and the external set A is conducted in Step 4, using the offspring population Y. More specifically, for each solution in Y, the neighbouring subproblems of its associated subproblem are updated in Steps 4a. Then, solution set generated by local search, L, is used to update P in Step 4b, as follows. For each solution $L_j \in L$, suppose it was generated from subproblem i. For each index $k \in B(i)$ of subproblem i, it replaces a neighbour x_k with L_j if L_j performs better than x_k with regard to subproblem k. Finally, the external set is updated in Step 4c, where the offspring population Y is merged with A and L; and the combined population Z is sorted by the fast-non-dominated-sorting and the crowding-distance-assignment. The best N solutions after sorting are kept in the external set as the new A.

5 Benchmark problems

In this paper, we consider two NP-hard multi-objective COPs: the multi-objective software next release problem (an instance of the knapsack problem, MONRP) and the multi-objective travelling salesman problem (MOTSP). These combinatorial optimization problems have been widely used in testing the performance of various MOEAs (Cai et al. 2012; Gaspar-Cunha 2005; Li and Landa-Silva 2011; Shim et al. 2012; Zhang et al. 2007).

5.1 The single and biobjective software next release problem

In the software next release problems, an existing software system is associated with several customers whose requirements must be considered for the next release of the software system. Let us assume the set of customers is denoted by

$$S = \{s_1, s_2, \dots, s_n\}$$

The set of all the requirements that need to be considered is denoted by

$$R = \{r_1, r_2, \dots, r_q\}$$

Resources, i.e., the cost of development, are required to accomplish each requirement. A cost vector can be used to denote the resources as follows:

$$C = (c_1, \dots, c_j, \dots, c_q),$$

where c_j denotes the cost of implementing a requirement $r_j \in R$. A request $q_{ij} \in Q$ denotes whether customer s_i requests requirement r_j to be implemented in the next release of the software; $q_{ij} = 1$ denotes that s_i requests r_j, and $q_{ij} = 0$ denotes otherwise.

Once customers' requirements have been satisfied in the next release of the software, the software provider can gain the certain income, denoted by

$$W = (w_1, \dots, w_i \dots, w_n),$$

where w_i denotes the income from customer s_i when the customers' requirements have been satisfied.

The task of the single-objective NRP is to find an optimal solution $X^\star$, to maximize $\text{Profit}(X) = \sum_{(i,1)\in X} w_i$ gained from customers, which is the sum of income gained from the selected customers, minus $\text{Cost}(X) \le b$, where b is a predefined budget constraint and $\text{Cost}(X) = \sum_{r_k \in R(X)} c_k$, where $R(X) = \bigcup_{(i,1)\in X, q_{ij}=1} \{r_j\}$ is the requirement for X.

It is very convenient to transform a single-objective NRP into a bi-objective NRP. The two objectives are the $\text{Profit}(X)$ to be maximized and the required $\text{Cost}(X)$ to be minimized. To apply MOEAs to this problem, we change the first objective to a minimization formulation.

5.2 The multi-objective travelling salesman problems

The travelling salesman problem (TSP) can be modelled as a graph G(V, E), where V ={1,…,n} is the set of vertices (cities) and $E = \{e_{i,j}\}_{n\times n}$ is the set of edges (connections between cities). The goal is to find a Hamiltonian cycle of minimal distance that visits each vertex only once. In the case of the multi-objective TSP (MOTSP), each edge $e_{i,j}$ is associated with multiple values such as cost, length, travelling time, etc. Each of them corresponds to one criterion. The mathematical definition of a MOTSP is as follows:

$$\text{minimize} \quad f_i(\pi) = \sum_{j=1}^{n-1} c^{(i)}_{\pi(j),\pi(j+1)} + c^{(i)}_{\pi(1),\pi(n)}$$
$$i = 1, \dots, m, \tag{8}$$

where $\pi = (\pi(1), \dots, \pi(n))$ is a permutation of cities and $c^{(i)}_{s,t}$ is the cost of the edge between city s and city t with respect to criterion i.

5.3 Test instances

A MONRP test instance with n customers and q requirements is denoted as Cu-n/R-q in this paper. Four randomly generated test instances are used in our studies, which include

Table 1 Details of the real NRP instances Xuan et al. (2012)

	Eclipse				Gnome			
Instance group name Source repository	nrp-e1	nrp-e2	nrp-e3	nrp-e4	nrp-g1	nrp-g2	nrp-g3	nrp-g4
Bug report ID	150001–160000		160001–170000		450001–460000		460001–470000	
Bug reports time period	Jul.2006–Oct.2006		Oct.2006–Jan.2007		Jun.2007–Jul.2007		Jul.2007–Aug.2007	
Requests of customer	4–20	5–30	4–15	5–20	4–20	5–30	4–15	5–20
Requirements	3502	4254	2844	3186	2690	2650	2512	2246
Cost of requirement	1–7	1–7	1–7	1–7	1–7	1–7	1–7	1–7
Customers	536	491	456	399	445	315	423	294
Profit of customer	10–50	10–50	10–50	10–50	10–50	10–50	10–50	10–50

Cu-200/R-1000, Cu-300/R-1500, Cu-400/R-2000 and Cu-500/R-2500. For the test instances of MONRP, the cost of each requirement ranges from 1 to 9 while the profit of implementing each requirement ranges from 5 to 50.

A MOTSP with n cities and m objectives is denoted as c-n-m in this paper. In our studies, we consider four randomly generated test instance of MOTSP with two objectives and two test instances of MOTSP with three objectives, which include c-300-2, c-400-2, c-500-2, c-600-2, c-200-3 and c-300-3.

In addition, real instances of MONRP have also been adopted. These real NRP test instances were mined from bug repositories of two open source software projects, namely Eclipse (2011) and Gnome (2011). The XML form of these bug repositories are available in Mining Challenges 2007 and 2009 of the IEEE Working Conference on Mining Software Repositories (MSR) (Mining challenges 2009; Xuan et al. 2012). The detailed descriptions of the real test instances of MONRP are summarized in Table 1. For convenience, test instances from the open source software projects of Eclipse (2011) and Gnome (2011) are denoted as $nrp-e$ and $nrp-g$, respectively, in this paper.

6 Experimental studies and discussions

To explore the performance of gMOMA-SA, uMOMA-SA and aMOMA-SA and understand their behaviour, this section describes the following experimental work:

- investigations of the effects of the local search, gMOMA-SA.
- investigations of the effects of two adaptive local searches, uMOMA-SA and aMOMA-SA.
- comparisons of gMOMA-SA, uMOMA-SA and aMOMA-SA with three single-objective and four multi-objective approaches on MONRP using real instances.

In our experiments, each algorithm is run independently 20 times on each test instance.

6.1 Performance indicators

The performance of a MOEA is usually evaluated in two aspects. First, the non-dominated set obtained should be as close to the true Pareto front as possible. This aspect is usually called convergence. Second, the non-dominated set obtained should be distributed diversely and uniformly. This aspect is usually called diversity. Various metrics have been designed to reflect either one or both aspects to evaluate the performance of a MOEA (Knowles et al. 2006). In this paper, we adopt the two best-known performance indicators, both of which are able to reflect performance in the two aspects at the same time, as follows:

- Hypervolume indicator (I_H) (Zitzler and Thiele 1999): Let $y^* = (y_1^*, \ldots, y_m^*)$ be a point in the objective space which is dominated by any of the Pareto optimal objective vectors. Let P be the approximation to the PF obtained in the objective space. Then the I_H value of P (with respect to y^*) is the volume of the region which is dominated by P and bounded by y^*. The higher the hypervolume, the better the approximation.
  In our experiments, unfortunately we don't know the maximum value of the actual Pareto front, so we set the maximum value of the obtained non-dominated set as the ideal maximum point: $y^* = (f_1^{\max}, f_2^{\max})$ for bi-objective test instances and $y^* = (f_1^{\max}, f_2^{\max}, f_3^{\max})$ for three-objective ones.
- Inverted generational distance (IGD) (Coello Coello and Cortés 2005): The IGD measures the average distance from a set of reference points P^* in the PF to the approximation set P. It can be formulated as follows:

$$\mathrm{IGD}(P, P^*) = \frac{1}{|P^*|} \sum_{v \in P^*} \mathrm{dist}(v, P), \tag{9}$$

where dist(*,*) is the Euclidean distance. Ideally, the points in P^* should be uniformly distributed on the PF. However, the true PF is not known in either the MONRP or the MOTSP test instances. In our experiments, P^* for a test instance is the set of all non-dominated solutions obtained by all the algorithms in all runs.

6.2 Experimental setups

All the experiments were conducted on a PC with a 3.4-GHz Intel Core $i7$ processor, 4 GB of RAM. All the algorithms were implemented in Matlab. Their parameter settings for MONRP and MOTSP are listed in Table 2. The algorithms compared (NSGA-II and MOEA/D) were fine-tuned and the parameters of MOMA-SA(s) (gMOMA-SA, uMOMA-SA and aMOMA-SA) were set in such a way that they shared the same key parameter values with MOEA/D.

The experimental setups are specified in Table 2, where the population sizes for all the compared algorithms were set to 200 for MONRP, 100 for bi-objective TSP and 120 for tri-objective TSP. The number of neighbours was set to 10 for MONRP and MOTSP. The crossover rate was set to 1 and mutation rate was set to $1/n$, where n is the length of a solution. For the MOTSP instances, each candidate solution was coded as a permutation, and the position based-crossover and exchange mutation operators (Larrañaga et al. 1999) were used for generating new solutions. Different from other compared algorithms, where the crossover rate was set to 1; the crossover rate for NSGA-II was set to 0.8 as the experimental analysis shows this setting was better than 1. The number of function evaluations (FEs) was set to 300,000 for all synthetic MONRP instances and 600,000 for all MOTSP instances. For real instances of MONRP, the number of function evaluations was set to 100,000.

The setting of N weight vectors $(\lambda^1, \ldots, \lambda^N)$ was controlled by a positive integer parameter H, which specifies the granularity or resolution of weight vectors, as in Zhang and Li (2007). Each individual weight takes a value from

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

The number of weight vectors was determined by both parameter H and the number of objectives m: $N = C_{H+m-1}^{m-1}$. H was set to 200 for bi-objective NRP, 100 for bi-objective TSP and 14 for 3-objective TSP, to ensure that the number of weight vectors was equal to the population size in Table 2.

We conducted experiments to investigate the sensitivity of parameter J, which is the number of generated local offspring solutions based on each parent solution, as shown in Fig. 1. They showed that the MOMA-SA has the best performance when the parameter J is set to around 5, which is the value we adopted in the other experiments. Simulated annealing was used as the local search meta-heuristic in our framework; its initial temperature (*Te*) was set to 100 and scheduling factor (α) was set to 0.99.

6.3 The effects of the local search (gMOMA-SA)

gMOMA-SA adopts simulated annealing as local search process; thus, it is necessary to verify its effects. Figs. 2 and 3 present the final solution sets with the best (i.e., largest) hypervolume values obtained by gMOMA-SA, NSGA-II, MOEA/D and MOEA/DD over 20 runs on bi-objective MONRP and MOTSP and 3-objective MOTSP instances. By MOEA/DD, we mean gMOMA-SA in which there is no local search. From another angle, MOEA/DD can also be considered as MOEA/D with the external set serving as the final output.

We make the following observations:

- For bi-objective MONRP instances, gMOMA-SA produces solutions with the best convergence and diversity.
- For bi-objective MOTSP instances, gMOMA-SA produces the set of solutions with the best convergence and which completely dominates the non-dominated sets produced by the other algorithms, although the non-dominated sets produced by all four of the algorithms compared have very good diversity.
- For 3-objective MOTSP instances, gMOMA-SA produces solutions with the best diversity, although all the

Table 2 Parameter settings

	Population size		# of neighbours		Crossover rates		Mutation rates		# of function evaluations	
	MONRP	MOTSP	MONRP	MOTSP	MONRP	MOTSP	MONRP	MOTSP	MONRP	MOTSP
NSGA-II	200	100 or 120	–	–	0.8	0.8	$1/n$	$1/n$	300,000	600,000
MOEA/D	200	100 or 120	10	10	1	1	$1/n$	$1/n$	300,000	600,000
gMOMA-SA	200	100 or 120	10	10	1	1	$1/n$	$1/n$	300,000	600,000
uMOMA-SA	200	100 or 120	10	10	1	1	$1/n$	$1/n$	300,000	600,000
aMOMA-SA	200	100 or 120	10	10	1	1	$1/n$	$1/n$	300,000	600,000

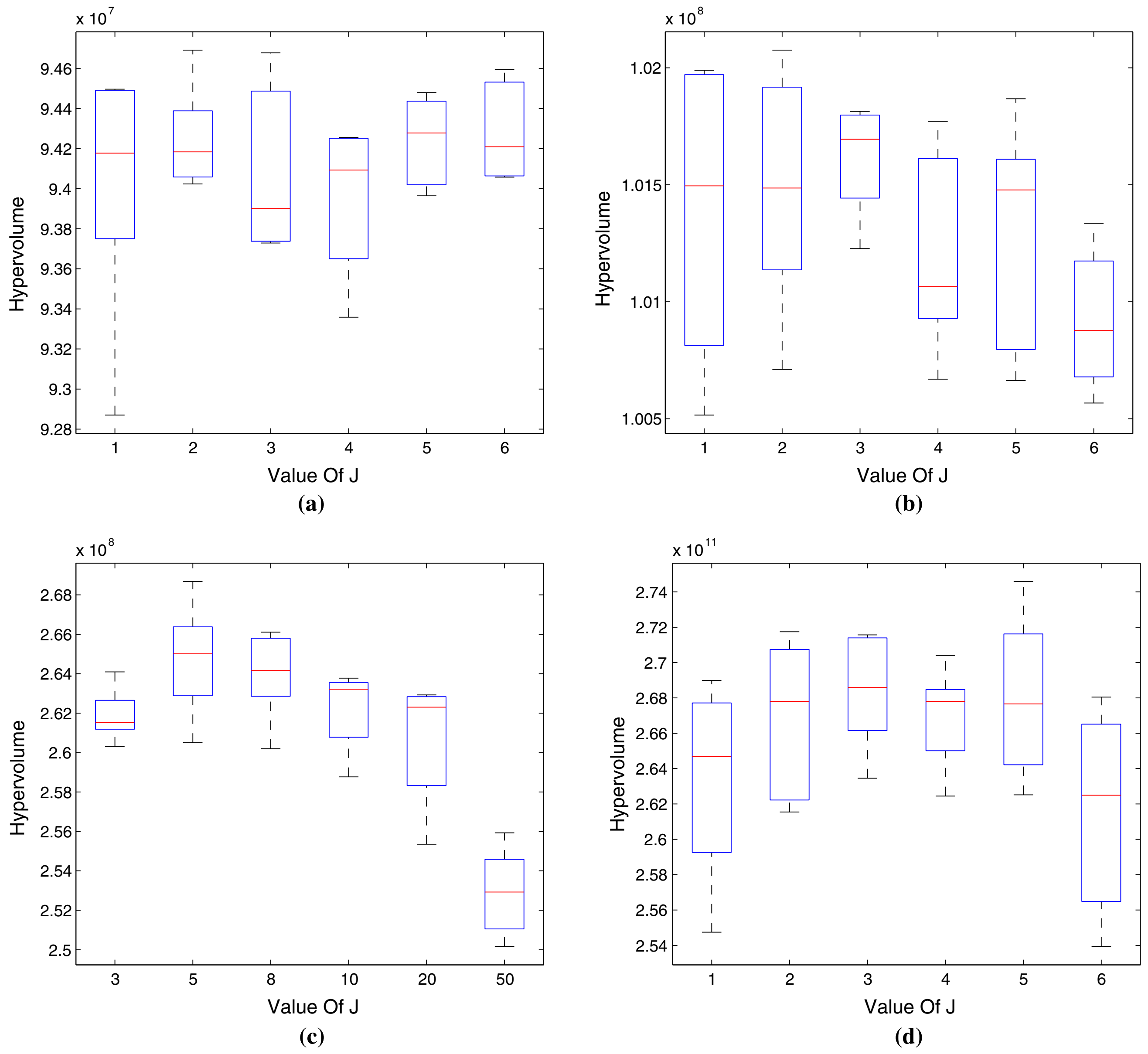


Fig. 1 Sensitivity test of parameter J (intensity of local search)

non-dominated sets produced by all four algorithms compared have very good convergence.

– Overall, it is clear that gMOMA-SA gives the best performance among all the approaches compared on all test instances.

The evolution of the average hypervolume values with the numbers of function evaluations in the four algorithms on MONRP test instances are plotted in Fig. 4. This figure shows both the convergence speed of each algorithm and the quality of its final solution set. It is worth noting that one local search makes small changes to the solutions and does not need a complete function evaluation. However, to compare gMOMA-SA with the algorithms without local searches (i.e., NSGA-II, MOEA/D and MOEA/DD) and show its efficiency, the function evaluations for either global or local searches are treated the same in Fig. 4.

To validate the results statistically, Table 3 lists the mean and standard deviation values of I_H on ten MONRP and MOTSP instances. Note that a larger I_H indicates better performance of the algorithm. Clearly, these results confirm that gMOMA-SA outperforms the other three algorithms on all ten test instances. All the above observations support the idea that using simulated annealing as a local search operator significantly improves the algorithmic performance.

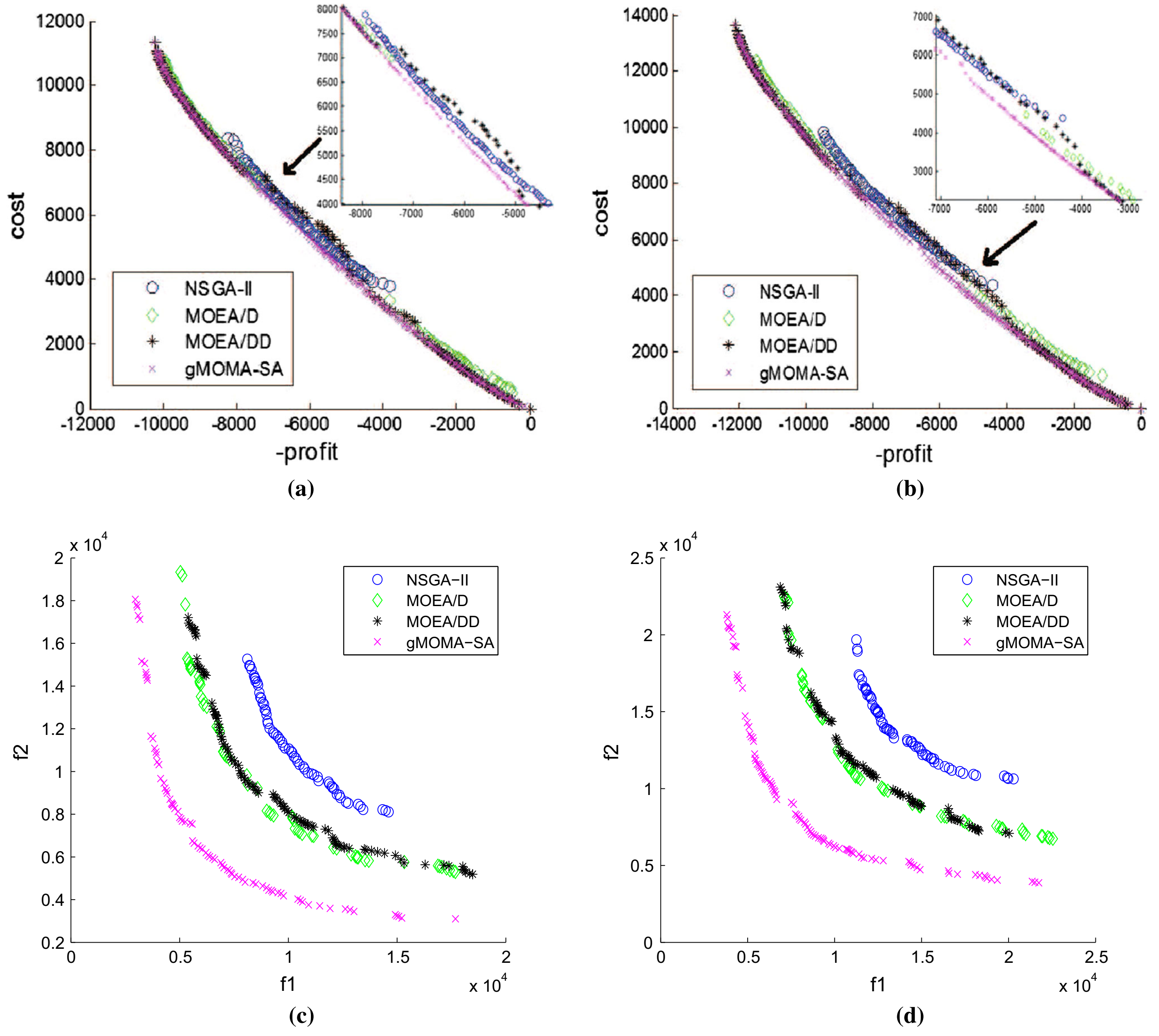


Fig. 2 The non-dominated solutions with the best (i.e., largest) hypervolume values obtained by NSGA-II, MOEA/D, MOEA/DD and gMOMA-SA over 20 runs, on MONRP and MOTSP instances

6.4 The effects of two adaptive local searches

As elaborated in Sect. 4.3, three different mechanisms of solution selection for local search were explored, namely gMOMA-SA, uMOMA-SA and aMOMA-SA.

Among them, gMOMA-SA selects each solution for each subproblem to conduct local search. uMOMA-SA and aMOMA-SA both use adaptive strategies to select solutions for local search. uMOMA-SA adopted the concept of utility proposed in Zhang and Rockett (2009), which uses the convergence information of subproblems to adaptively decide which solutions to choose for local search. In contrast, aMOEA-SA uses both the convergence and diversity information extracted from the dominance archive to adaptively select solutions for local search.

To understand the effects of the various methods of selecting points for adaptive local searches, we conducted the following comparisons:

6.4.1 gMOMA-SA vs. uMOMA-SA

Table 4 compares the performance of gMOMA-SA and uMOMA-SA on ten MONRP and MOTSP instances. Clearly, in terms of I_H, uMOMA-SA outperforms gMOMA-SA on all six MOTSP instances with statistical significance and on all four MONRP instances, but not always at the level

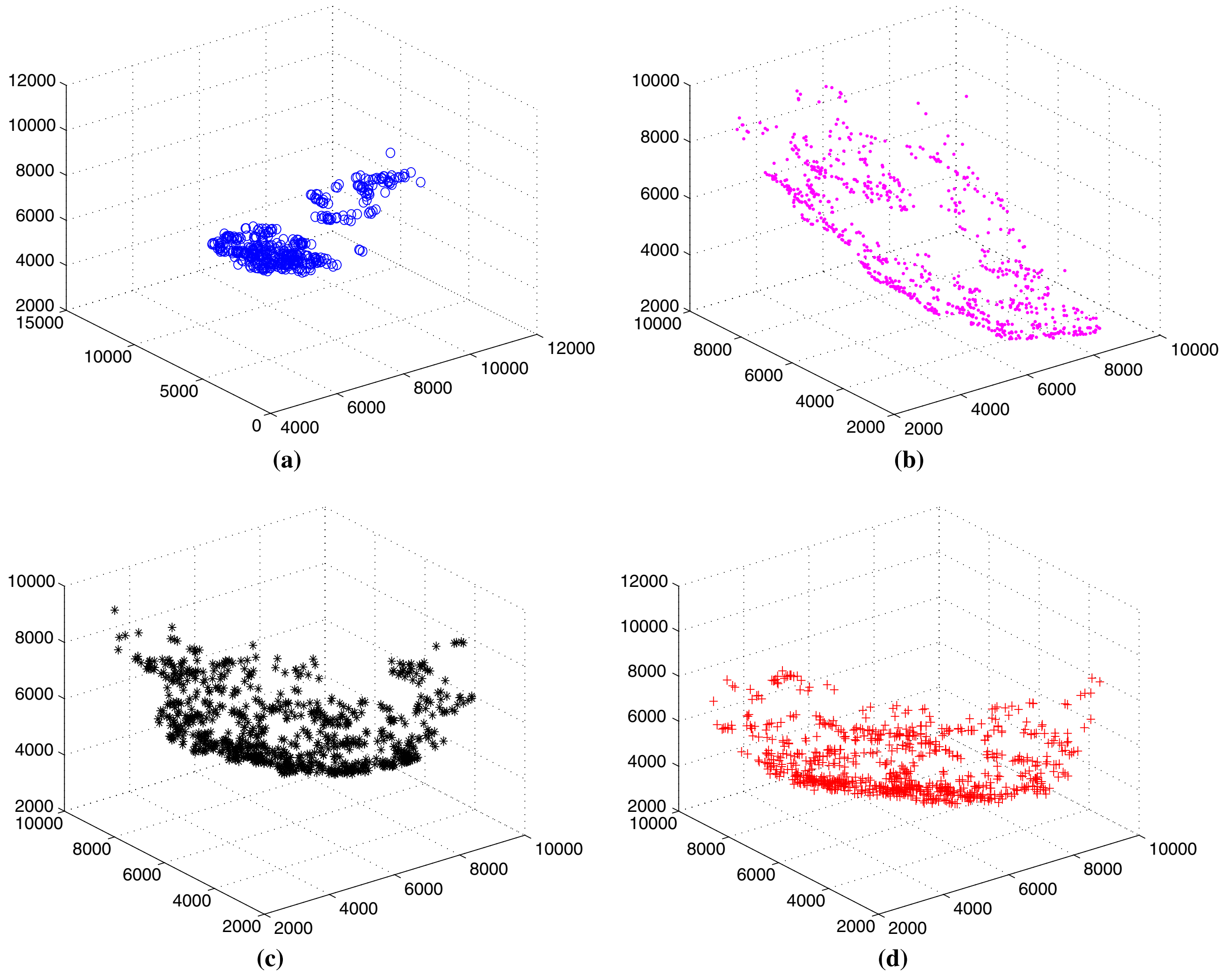


Fig. 3 The non-dominated solutions with the best (i.e., largest) hypervolume values obtained by NSGA-II, MOEA/D, MOEA/DD and gMOMA-SA over 20 runs, on MOTSP instance c-200-3

of statistical significance of $p < 0.05$. In terms of IGD, uMOMA-SA outperforms gMOMA-SA on all MONRP and MOTSP instances except c-300-3. However, statistical significance can be seen only on Cu-500/R-2500, c-400-2 and c-500-2. Based on these results, we can claim that uMOMA-SA is better than gMOMA-SA. This is consistent with our argument in Sect. 1, that is, that utility-based solution selection for local search is able to effectively utilize each subproblem's convergence information to guide local search during the optimization process.

6.4.2 gMOMA-SA vs. aMOMA-SA

Table 5 compares the performances of gMOMA-SA and aMOMA-SA, in terms of both I_H and IGD metric. In terms of I_H, it can be observed that aMOMA-SA performs statistically better than gMOMA-SA, on all ten instances except for c-200-3. On c-200-3, aMOMA-SA produces a larger I_H value; however, it is not statistically significant. In terms of IGD, aMOMA-SA performs statistically better than gMOMA-SA on all ten instances except for Cu-200/R-1000 and c-200-3. Based on these results, we can claim that aMOMA-SA performs better than gMOMA-SA.

6.4.3 uMOMA-SA vs. aMOMA-SA

Because uMOMA-SA and aMOMA-SA adopt two different adaptive local search strategies, it is interesting to study the commonalities and differences between them. For this purpose, we have run uMOMA-SA and aMOMA-SA with the same parameter settings in Table 2 on all the test instances.

Table 6 presents the average values of I_H and IGD over 20 independent runs. In terms of I_H, it can be observed that aMOMA-SA performs statistically better than gMOMA-SA, on all ten instances except for c-200-3. On c-200-3, aMOMA-SA produces a larger I_H value; however, it is

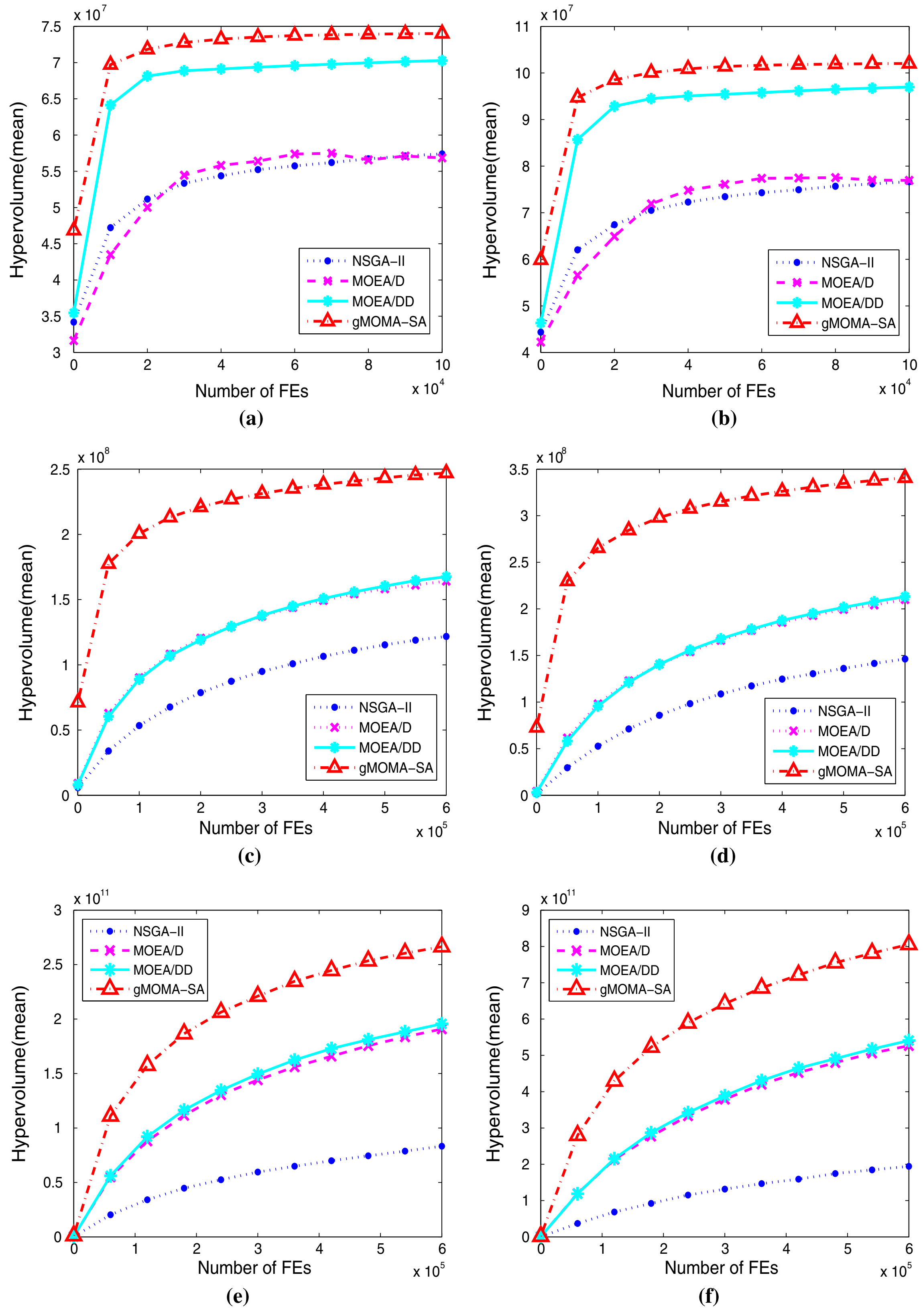


Fig. 4 Convergence plots in terms of average I_H over 20 runs, obtained by four algorithms on MONRP and MOTSP instances

Table 3 Mean and standard deviation values of I_H obtained by four algorithms on MONRP and MOTSP instances

	NSGA-II	MOEA/D	MOEA/DD	gMOMA-SA
Cu-200/R-1000				
Mean	1.6518e+07	1.7793e+07	1.8665e+07	**1.8732e+07**
Std	1.9904e+05	1.0193e+05	3.7785e+04	2.5803e+04
Cu-300/R-1500				
Mean	3.1568e+07	2.6528e+07	3.5676e+07	**3.6332e+07**
Std	4.3181e+05	4.0284e+06	3.4274e+05	6.3052e+05
Cu-400/R-2000				
Mean	6.0050e+07	6.2748e+07	7.1928e+07	**7.3481e+07**
Std	8.7525e+05	1.1012e+06	5.1543e+05	5.2123e+05
Cu-500/R-2500				
Mean	7.6671e+07	7.6933e+07	9.6964e+07	**1.0136e+08**
Std	1.4111e+06	2.5817e+06	1.0428e+06	4.3044e+05
c-300-2				
Mean	7.8647e+07	1.0028e+08	1.0172e+08	**1.1757e+08**
Std	1.0299e+06	1.0677e+06	1.4351e+06	1.2675e+06
c-400-2				
Mean	1.2177e+08	1.6419e+08	1.6759e+08	**1.9890e+08**
Std	2.2033e+06	4.1758e+06	1.9895e+06	1.8527e+06
c-500-2				
Mean	1.4621e+08	2.1001e+08	2.1324e+08	**2.6294e+08**
Std	2.8420e+06	2.7892e+06	3.6784e+06	3.5585e+06
c-600-2				
Mean	3.0337e+08	4.4358e+08	4.4479e+08	**5.3350e+08**
Std	5.7284e+06	5.9462e+06	5.0039e+06	3.8984e+06
c-200-3				
Mean	8.3293e+10	1.9088e+11	1.9563e+11	**2.6648e+11**
Std	4.9150e+09	5.5013e+09	7.1752e+09	5.7416e+09
c-300-3				
Mean	1.9450e+11	5.2645e+11	5.4080e+11	**8.0558e+11**
Std	1.6676e+10	1.5060e+10	1.7951e+10	1.5218e+10

The best mean values are highlighted in bold

not statistically significant. In terms of IGD, aMOMA-SA performs statistically better than gMOMA-SA on all ten instances except for Cu-200/R-1000 and c-200-3. Based on these results, we can claim that aMOMA-SA performs better than uMOMA-SA.

The above observations support the assertion that the external-set-guided selection uses not only convergence information but also diversity information to effectively guide the search process during the whole optimization process, which further enhances the performance compared with uMOMA-SA.

To further understand two adaptive mechanisms in uMOMA-SA and aMOMA-SA, we conducted two sets of experiments to show the conditions of solution selection for local search over the last 10 generations of the optimization process. For the first set of experiments, the utility values of each subproblem in uMOMA-SA are shown in the left panel of Fig. 5. For the second set of experiments, the contributions of subproblems to the external set are shown in the right panel of Fig. 5.

Two important phenomena can be observed in Fig. 5. First, different subproblems either have very different utility values or generate very different numbers of diverse non-dominated solutions. As our proposed memetic framework decomposes the objective space into multiple subproblems, the identification of promising regions is transformed into the identification of the promising subproblems. By estimating the convergence information using utility estimation or the contributions of various subproblems to the non-dominated archive, both adaptive mechanisms are able to identify the likelihood of promising regions in the objective space.

Second, although uMOMA-SA and aMOMA-SA use different strategies to select solutions for local search, they share similar patterns in terms of frequencies of each subproblem to

Table 4 Wilcolxon's rank sum at 0.05 significance level and comparison between gMOMA-SA and uMOMA-SA in term of I_H and IGD on MONRP and MOTSP instances

Instance	I_H		Wilcoxon test	IGD		Wilcoxon test
	gMOMA-SA	uMOMA-SA	*p* value	gMOMA-SA	uMOMA-SA	*p* value
Cu-200/R-1000						
Mean	1.8732e+07	**1.8732e+07**	0.5979	14.6043	**14.2365**	0.4094
Std	2.5803+04	1.6518e+04		1.5322	1.3284	
Cu-300/R-1500						
Mean	3.6332e+07	**3.6152e+07**	0.3104	91.4978	**93.4667**	0.6168
Std	6.3052e+05	4.5845e+05		17.8290	14.7129	
Cu-400/R-2000						
Mean	7.3481e+07	**7.3595e+07**	0.7972	84.6500	**79.0654**	0.5609
Std	5.2123e+05	3.3390e+05		18.2419	12.6135	
Cu-500/R-2500						
Mean	1.0136e+08	**1.0159e+08**	0.0962	93.5148	**82.8278**	**0.0179**
Std	4.3044e+05	4.0189e+05		17.3351	10.7403	
c-300-2						
Mean	1.1757e+08	**1.1860e+08**	**0.0114**	475.6433	**473.8120**	0.9031
Std	1.2675e+06	1.0999e+06		112.4731	109.4998	
c-400-2						
Mean	1.9890e+08	**2.0263e+08**	**6.6737e−06**	736.9799	**583.2016**	**9.2091e−04**
Std	1.8527e+06	2.0353e+06		138.6911	130.1067	
c-500-2						
Mean	2.6294e+08	**2.6966e+08**	**4.5401e−06**	945.7849	**718.2724**	**8.3572e−04**
Std	3.5585e+06	2.8175e+06		204.9886	207.3562	
c-600-2						
Mean	5.3350e+08	**5.3863e+08**	**6.6104e−05**	1.0461e+03	**1.0167e+03**	0.5979
Std	3.8984e+06	3.1461e+06		215.7218	213.3891	
c-200-3						
Mean	2.6648e+11	**2.7333e+11**	**4.1550e−04**	631.7064	**621.6250**	0.5428
Std	5.7416e+09	5.1255e+09		77.7115	101.7903	
c-300-3						
Mean	8.0558e+11	**8.2858e+11**	**1.2941e−04**	**941.6841**	1.0216e+03	0.0720
Std	1.5218e+10	1.6313e+10		104.5864	162.9990	

The values in bold are significantly better than other algorithms

be selected for local search. This can be observed by comparing the utility values of subproblems (the right panel of Fig. 5 with the contributions of subproblems to the dominance archive (the left panel of Fig. 5), both of which determine the probabilities that each subproblem is selected for local search.

6.5 Analysis of computational time

Table 7 presents the average computational time of each algorithm for one run on MOTSP and MONRP. It can be seen from Table 7 that MOEA/D runs much faster than NSGA-II. On the other hand, the computational time of MOEA/DD is very close to that of NSGA-II, which reflects that a large portion of the computational time spent on MOEA/DD is for NSGA-II selection. Furthermore, the computational time of our proposed algorithms (gMOMA-SA, uMOMA-SA and aMOMA-SA) is just slightly larger than MOEA/D but far less than NSGA-II and MOEA/DD. This indicates that our proposed algorithm is very efficient by using local search. In addition, it is worth noting that we have used Matlab to implement all the algorithms. The computational time can be further improved using C++.

6.6 Single-objective vs. multi-objective optimization algorithms on real NRP instances

In this section, we compare various algorithms for solving NRP problems based on real instances. These real NRP

Table 5 Wilcolxon's rank sum at 0.05 significance level and comparison between gMOMA-SA and aMOMA-SA in term of I_H and IGD on MONRP and MOTSP instances

Instance	I_H		Wilcoxon test	IGD		Wilcoxon test
	gMOMA-SA	aMOMA-SA	p value	gMOMA-SA	aMOMA-SA	p value
Cu-200/R-1000						
Mean	1.8732e+07	**1.8743e+07**	0.0663	14.6043	**14.5498**	0.9892
Std	2.5803+04	1.4609e+04		1.5322	1.9024	
Cu-300/R-1500						
Mean	3.6332e+07	**3.7184e+07**	**1.8935e−05**	91.4978	**64.4330**	**1.2505e−05**
Std	6.3052e+05	4.2604e+05		17.8290	15.1879	
Cu-400/R-2000						
Mean	7.3481e+07	**7.4301e+07**	**5.2269e−07**	84.6500	**49.8014**	**3.4156e−07**
Std	5.2123e+05	2.2610e+05		18.2419	10.5886	
Cu-500/R-2500						
Mean	1.0136e+08	**1.0239e+08**	**2.5629e−07**	93.5148	**55.1305**	**6.9166e−07**
Std	4.3044e+05	2.9678e+05		17.3351	12.4397	
c-300-2						
Mean	1.1757e+08	**1.2200e+08**	**7.8980e−08**	475.6433	**281.4687**	**3.4995e−06**
Std	1.2675e+06	1.5036e+06		112.4731	80.0110	
c-400-2						
Mean	1.9890e+08	**2.0758e+08**	**6.7596e−08**	736.9799	**390.1924**	**3.4156e−07**
Std	1.8527e+06	2.3867e+06		138.6911	103.1213	
c-500-2						
Mean	2.6294e+08	**2.7813e+08**	**6.5756e−08**	945.7849	**466.2101**	**8.6848e−07**
Std	3.5585e+06	2.8940e+06		204.9886	107.1382	
c-600-2						
Mean	5.3350e+08	**5.5480e+08**	**6.0426e−08**	1.0461e+03	**590.7146**	**9.1573e−06**
Std	3.8984e+06	3.7734e+06		215.7218	154.6201	
c-200-3						
Mean	2.6648e+011	**2.7534e+11**	**4.6804e−05**	631.7064	**589.7356**	0.4301
Std	5.7416e+09	5.6315e+09		77.7115	62.1813	
c-300-3						
Mean	8.0558e+11	**8.4217e+11**	**3.4995e−06**	941.6841	**876.4832**	**0.0491**
Std	1.5218e+10	1.9663e+10		104.5864	121.7361	

The values in bold are significantly better than other algorithms

instances were mined from bug repositories of two open source software projects (Xuan et al. 2012), namely Eclipse (2011) and Gnome (2011).

In previous literature, NRP is defined as a constrained single-objective optimization problem, that is, the income from customers is to be maximized under various budget (cost) bounds. The budget bounds can be defined by the users based on the cost ratio, which is the budget bounds over the total costs when all requirements are implemented. It is very natural to reformulate NRP into a bi-objective optimization problem by converting the cost constraints into another objective.

Three single- and six multi-objective algorithms were tested and compared on all the real NRP instances. The single-objective optimization algorithm include genetic algorithm (GA) (Sagrado et al. 2010), multi-start simulated annealing (MSSA) (Mart 2003) and the Backbone-based Multilevel Algorithm (BMA) (Xuan et al. 2012); and the multi-objective EAs compared are NSGA-II (Deb et al. 2002), MOEA/D (Zhang and Li 2007), MOEA/DD (Cai and Wei 2013), MOEA/D-STM (Li et al. 2014), gMOMA-SA, uMOMA-SA and aMOMA-SA. The parameter setting of MOEA/D-STM is the same as that of MOEA/D, as shown in Table 2. To conduct fair comparisons, we set the number of function evaluations for all the compared algorithms, single- or multi-objective, to 100,000.

Table 8 shows the best and the average values of profit of the best solutions over 30 runs, obtained by ten approaches,

Table 6 Wilcolxon's rank sum at 0.05 significance level and comparison between uMOMA-SA and aMOMA-SA in term of I_H and IGD on MONRP and MOTSP instances

Instance	I_H		Wilcoxon test	IGD		Wilcoxon test
	uMOMA-SA	aMOMA-SA	p value	uMOMA-SA	aMOMA-SA	p value
Cu-200/R-1000						
Mean	1.8732e+07	**1.8743e+07**	**0.0239**	**14.2635**	14.5498	0.3235
Std	1.6518+04	1.4609e+04		1.3284	1.9024	
Cu-300/R-1500						
Mean	3.6152e+07	**3.7184e+07**	**2.0616e−06**	93.4667	**64.4330**	**2.0407e−05**
Std	4.5845e+05	4.2604e+05		14.7129	15.1879	
Cu-400/R-2000						
Mean	7.3595e+07	**7.4301e+07**	**1.3761e−06**	79.0654	**49.8014**	**4.5390e−07**
Std	3.3390e+05	2.2610e+05		12.6135	10.5886	
Cu-500/R-2500						
Mean	1.0159e+08	**1.0239e+08**	**2.6898e−06**	82.8278	**55.1305**	**9.1266e−07**
Std	4.0190e+05	2.9678e+05		10.7403	12.4397	
c-300-2						
Mean	1.1860e+08	**1.2200e+08**	**1.9177e−07**	473.8120	**281.4687**	**2.0616e−06**
Std	1.0999e+05	1.5036e+06		109.4998	80.0110	
c-400-2						
Mean	2.0263e+08	**2.0758e+08**	**1.5757e−06**	583.2016	**390.1924**	**1.8074e−05**
Std	2.0353e+06	2.3867e+06		130.1067	103.1213	
c-500-2						
Mean	2.6966e+08	**2.7813e+08**	**3.4156e−07**	718.2724	**466.2101**	**9.2780e−05**
Std	2.8175e+06	2.8940e+06		207.3562	107.1382	
c-600-2						
Mean	5.3863e+08	**5.5480e+08**	**1.4149e−05**	1.0167e+03	**590.7146**	**1.0473e−06**
Std	3.1461e+06	3.7734e+06		213.3891	154.6201	
c-200-3						
Mean	2.7333e+11	**2.7534e+11**	0.2977	621.6250	**589.7356**	0.5792
Std	5.1255e+09	5.6315e+09		101.7903	62.1813	
c-300-3						
Mean	8.2858e+11	**8.4217e+11**	**0.0223**	1.0216e+03	**876.4832**	**0.0033**
Std	1.6313e+10	1.9663e+10		162.9990	121.7361	

The values in bold are significantly better than other algorithms

on eight real NRP instances. Each NRP instance may have two different cost ratios (the ratio of the budget to the maximum cost, equal to either 0.3 or 0.5). For convenience, the nrp-g1 instance with cost ratio 0.3 is denoted as nrp-g1-0.3, for example, as shown in the first column of Table 8. The budget bounds corresponding to the cost ratio are presented in the second column of the table.

From Table 8, it can be seen that the performances of all of the multi-objective optimization algorithms, except for NSGA-II, were superior to those of the single-objective optimization algorithms, in terms of finding solutions with higher profit values within predefined budget bounds. Although BMA in Xuan et al. (2012) achieves the best performance in terms of average and best profit values for all the instances among the single-objective approaches, its performance is worse than MOEA/D, MOEA/DD, MOEA/D-STM, gMOMA-SA, uMOMA-SA and aMOMA-SA in all the test instances, except for nrp-g2 and nrp-g4-0.3. This observation is interesting given the naive intuition that multi-objective problems may be more difficult to solve than single-objective problems. One possible explanation is that the multi-objective formulation of NRP constructs a different search space from that of the single-objective formulation, which may help the search process circumvent local optima. In addition, it is clear to see from Table 8 that both of our proposed approaches (uMOMA-SA or aMOMA-SA) achieve the best performance in most NRP instances. It is worth noting that uMOMA-SA and aMOMA-SA outperform the

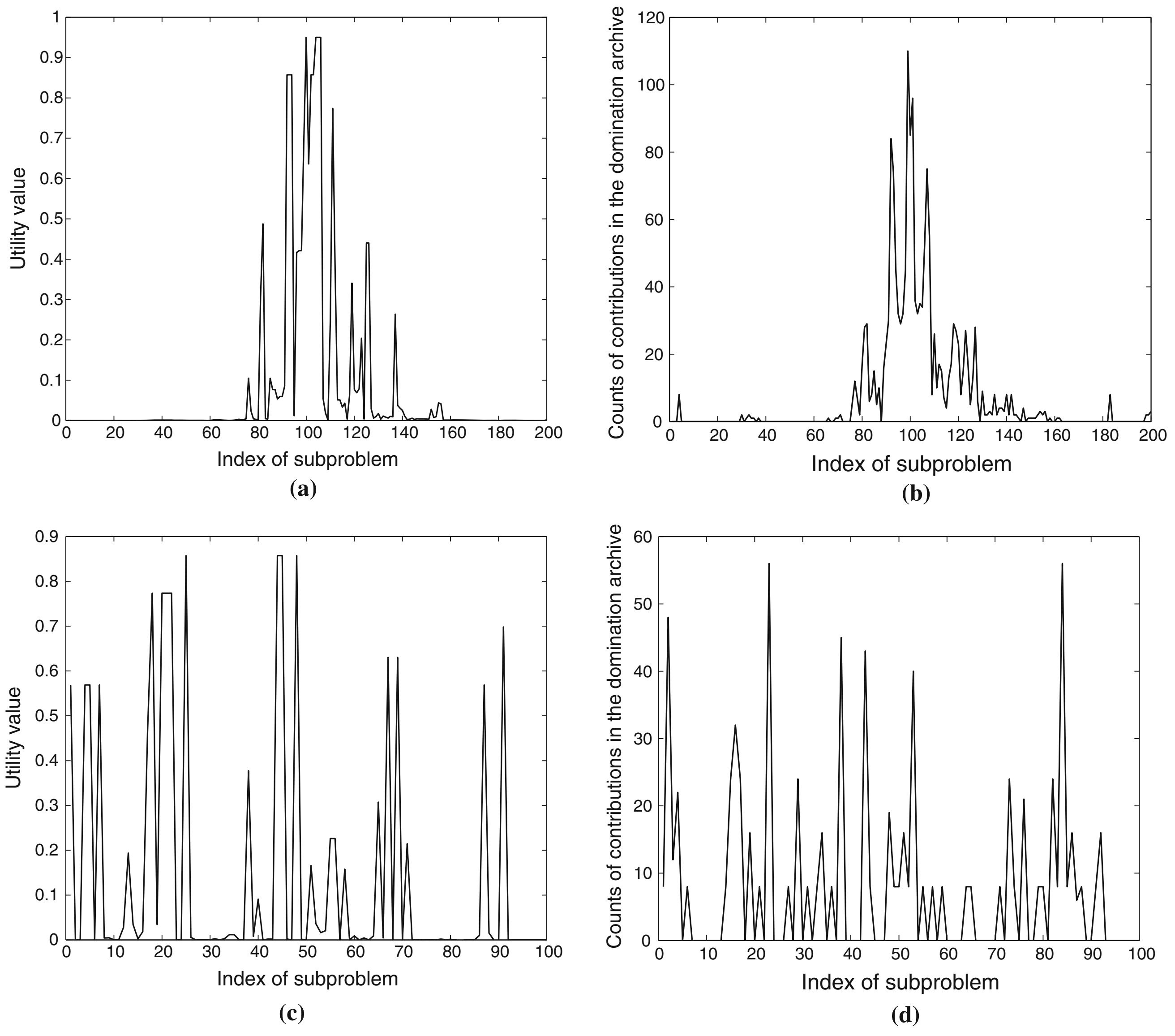


Fig. 5 Utility values in uMOMA-SA (the *left panel*) and the contributions of subproblems to the dominance archive in aMOMA-SA (the *right panel*) over the past 10 generations on MONRP and MOTSP instances

Table 7 Comparisons among three single-objective and six multi-objective optimization algorithms on real NRP instances

Instance	Time (s)					
	NSGA-II	MOEA/D	MOEA/DD	gMOMA-SA	uMOMA-SA	aMOMA-SA
Cu-200/R-1000	1366	313	1315	387	370	412
Cu-300/R-1500	1578	388	1543	400	458	504
Cu-400/R-2000	1725	403	1648	464	502	542
Cu-500/R-2500	1894	462	1795	542	562	586
c-300-2	2846	1263	1296	456	420	504
c-400-2	3182	2154	2634	567	528	645
c-500-2	4026	3035	4174	712	658	801
c-600-2	5638	4546	5500	1018	816	961
c-200-3	2344	881	2594	430	575	446
c-300-3	2573	1460	3131	528	582	611

Significantly best values are highlighted in bold

Table 8 Comparisons of three single-objective and six multi-objective optimization algorithms on real NRP instances

Instance			MSSA	GA	BMA	NSGA-II	MOEA/D	MOEA/DD	MOEA/D-STM	gMOM-SA	uMOMA-SA	aMOMA-SA
nrp-e1-0.3	Bound:3945	Best	5723	6662	7572	7135	7836	7878	7836	7773	7856	**7879**
		Average	5656.5	6553.4	7528.2	7063.6	**7774.6**	7812	7774.1	7715.8	7771.3	7772.3
nrp-e1-0.5	Bound:6575	Best	6575	9801	10664	10563	10949	10933	11014	11000	**11032**	11008
		Average	8550	9756.3	10589.2	10491.4	10881.3	10872.2	**10970.6**	10877.6	10893.4	10921.0
nrp-e2-0.3	Bound:4722	Best	5321	6275	7169	6699	7357	7344	7311	7290	7360	**7367**
		Average	5281.0	6219.6	7109.9	6472.6	7280.8	7277.6	7039.4	7231.5	7278.7	**7283.6**
nrp-e2-0.5	Bound:7871	Best	7932	9203	10098	9812	**10286**	10267	10244	10263	10278	10268
		Average	7869.6	9172.9	9999.8	9773.6	**10234.8**	10182.4	9905.6	10163.1	10205.3	10189.4
nrp-e3-0.3	Bound:4778	Best	5031	5795	6461	8558	8858	8854	8763	8797	8852	**8868**
		Average	4906.4	5693.1	6413.0	8504.3	7800.6	8795.1	8257.9	8673.3	**8803.9**	8789.1
nrp-e3-0.5	Bound:7964	Best	7436	8491	9175	10401	11922	11922	11921	11922	11921	**11926**
		Average	7340.5	8391.1	9100.1	10183.2	11538.8	11880.0	11818.4	11864.5	11876.4	**11887.4**
nrp-e4-0.3	Bound:5099	Best	4332	5065	5692	7251	7477	7480	7446	7435	**7488**	7474
		Average	4267.9	5023.8	5636.2	7209.8	6961.5	7322	7355.5	7209.8	7362.4	**7391.6**
nrp-e4-0.5	Bound:8499	Best	6459	7487	8043	9488	10176	10176	10175	10161	10175	**10178**
		Average	6391.1	7418.9	7968.0	9090.2	9866.6	10111.2	10094.4	10098.7	10095.4	**10141.9**
nrp-g1-0.3	Bound:4140	Best	4806	5494	5938	6036	6277	6281	6151	6278	6262	**6283**
		Average	4723.0	5437.0	5911.3	5933.7	6159.0	6126.6	5941.1	6155.3	6173.8	**6205.4**
nrp-g1-0.5	Bound:6900	Best	7339	8223	8714	8825	9123	9118	9114	9100	**9134**	9126
		Average	7250.3	8151.7	8660.0	8792.3	9003	9086.4	8964.6	9019.4	9050.4	**9088.6**
nrp-g2-0.3	Bound:3677	Best	3583	4256	**4526**	4361	4472	4475	4438	4468	4472	4474
		Average	3549.9	4195.5	**4486.2**	4309.6	4262.4	4430.1	4415	4432.3	4442.2	4444.8
nrp-g2-0.5	Bound:6129	Best	5433	6219	**6502**	6322	6398	6413	6413	6420	6407	6413
		Average	5359.8	6138.4	**6470.2**	6283.3	6238.3	6387.0	6264.3	6384.2	6385.1	6375.3
nrp-g3-0.3	Bound:4258	Best	4663	5351	5802	6372	6581	6573	6533	6560	**6593**	6584
		Average	4593.1	5296.6	5736.5	6317.6	6373.9	6527.1	6440.2	6492.3	6525.1	**6541.6**
nrp-g3-0.5	Bound:7097	Best	7032	7903	8714	9008	9375	**9375**	9333	9355	9370	9355
		Average	6948.1	7849.8	8326.8	8952.1	9272.2	**9336.3**	9252.8	9301.4	9328.4	9329.9
nrp-g4-0.3	Bound:3120	Best	3386	3951	**4190**	4029	4109	4113	4085	4106	4112	4121
		Average	3313.9	3909.9	**4159.0**	4017	4075.8	4068.5	4070.6	4078.6	4077.2	4078.0
nrp-g4-0.5	Bound:5350	Best	5041	5751	6030	5978	6054	6052	**6058**	6052	6056	6057
		Average	4991.6	5721.3	5986.5	5942.5	5994.3	6024.9	5935.6	6040.2	6040.7	**6046.6**

The bold values indicate the highest profits within the budget bounds obtained by all the algorithms for different NRP instances state-of-art MOEA/D-STM on all the test instances, except for nrp-g4-0.5 and nrp-e1-0.5.

7 Conclusion

In this paper, we have proposed a decomposition-based multi-objective memetic framework using two adaptive mechanisms for local search (called uMOMA-SA and aMOMA-SA) to address COPs. The first mechanism adaptively selects solutions for local search based on the solutions' convergence information (utility). The second adaptive mechanism extracts convergence and diversity information from an external dominance archive to guide the selection of promising solutions for local search. The effects of the two adaptive mechanisms were investigated on MONRP and MOTSP instances. In addition, we compared three single-objective and four multi-objective optimization approaches on software next release problems using real instances mined from a bug repository. The results showed that the multi-objective optimization approaches performed better than the single-objective ones overall and that aMOMA-SA had the best performances among all the approaches compared.

Acknowledgments This work was supported in part by the National Basic Research Program (also called 973 Program) of China under Grant 2014CB744903, by the National Natural Science Foundation of China (NSFC) under Grant 61300159 and 61175073, by the Natural Science Foundation of Jiangsu Province under Grant BK20130808, by China Postdoctoral Science Foundation under Grant 2015M571751 and by the Fundamental Research Funds for the Central Universities of China under Grant NZ2013306.

Compliance with ethical standards

Conflict of interest The author(s) of this publication has research support from Nanjing University of Aeronautics and Astronautics. The terms of this arrangement have been reviewed and approved by the university in accordance with its policy on objectivity in research.

References

Alsheddy A, Tsang E (201) Guided pareto local search based frameworks for biobjective optimization. In: IEEE congress on evolutionary computation (CEC), pp 1–8

Arroyo JEC, Armentano VA (2005) Genetic local search for multi-objective flowshop scheduling problems. Eur J Oper Res 167(3):717–738

Borges PC, Hansen MP (1998) A basis for future successes in multiobjective combinatorial optimization. Technical Report IMM-REP-1998-8. Institute of Mathematical Modelling, Technical University of Denmark

Cai Xinye, Wei Ou, Huang Zhiqiu (2012) Evolutonary approches for multi-objective next release problem. Comput Inf 31(4):847–875

Cai X, Li Y, Fan Z, Zhang Q (2014) An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization. IEEE Trans Evol Comput

Cai X, Wei O (2013) A hybrid of decomposition and domination based evolutionary algorithm for multi-objective software next release problem. In: 10th IEEE international conference on control and automation

Caponio A, Neri F (2009) Integrating cross-dominance adaptation in multi-objective memetic algorithms. Stud Comput Intell 171:325–351

Chang PC, Chen SH, Zhang Q, Lin JL (2008) MOEA/D for flowshop scheduling problems. In: 2008 congress on evolutionary computation (CEC'2008), Hong Kong. IEEE Service Center, pp 1433–1438

Chen X, Ong Y-S, Lim M-H, Tan KC (2011) A multi-facet survey on memetic computation. IEEE Trans Evol Comput 15(5):591–607

Coello Coello CA, Cortés NC (2005) Solving multiobjective optimization problems using an artificial immune system. Genet Program Evol Mach 6(2):163–190

Deb K (2001) Multi-objective optimization using evolutionary algorithms. Wiley, New York

Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197

del Sagrado J, del Aguila IM, Orellana FJ (2010) Ant colony optimization for the next release problem: a comparative study, pp 67–76

Droste S, Jansen T, Wegener I (2002) On the analysis of the (1+1) evolutionary algorithm. Theor Comput Sci 276(1–2):51–81

Durillo JJ, Zhang Y, Alba E, Harman M, Nebro Antonio J (2011) A study of the bi-objective next release problem. Empir Softw Eng 16(1):29–60

Eckart Z, Marco L, Lothar T (2002) SPEA2: improving the strength pareto evolutionary algorithm. In: Giannakoglou K, Tsahalis D, Periaux J, Papailou P, Fogarty T (eds) EUROGEN 2001, evolutionary methods for design, optimization and control with applications to industrial problems, Athens, pp 95–100

Eclipse (2011). http://www.eclipse.org/

García-Martínez C, Cordón O, Herrera F (2007) A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. Eur J Oper Res 180(1):116–148

Gaspar-Cunha A (2005) A multi-objective evolutionary algorithm for solving traveling salesman problems: application to the design of polymer extruders. In: Bernardete R, Albrecht RF, Andrej D, Pearson DW, Steele NC (eds) Adaptive and natural computing algorithms, Coimbra. Springer, pp 189–193

Gnome (2011). http://www.gnome.org/

Grosan C, Abraham A (2007) Hybrid evolutionary algorithms: methodologies, architectures, and reviews. In: Hybrid evolutionary algorithms. Springer, Berlin

Ishibuchi H, Yoshida T, Murata T (2003) Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling. IEEE Trans Evol Comput 7(2):204–223

Ishibuchi H, MurataT (1996) Multi-objective genetic local search algorithm. In: Fukuda T, Furuhashi T (eds) Proceedings of the 1996 international conference on evolutionary computation, Nagoya, IEEE, pp 119–124

Ishibuchi H, Sakane Y, Tsukamoto N, Nojima Y (2009) Adaptation of scalarizing funtions in moea/d: an adaptive scalarizing funtion-based multiobjective evolutionary algorithm. In: Ehrgott M, Fonseca CM, Gandibleux X, Hao J-K, Sevaux M (eds) Evolutionary multi-criterion optimization. 5th International Conference, EMO 2009, lecture notes in computer science, vol 5467, Nantes. Springer, pp 438–452

Jaszkiewicz A (2002) On the performance of multiple-objective genetic local search on the 0/1 knapsack problem–a comparative experiment. IEEE Trans Evol Comput 6(4):402–412

Kafafy A, Bounekkar A, Bonnevay S (2012) Hybrid metaheuristics based on moea/d for 0/1 multiobjective knapsack problems: a comparative study. In: IEEE congress on evolutionary computation, pp 1–8

Ke L, Zhang Q, Battiti R (2014) A simple yet efficient multiobjective combinatorial optimization method using decompostion and Pareto local search. IEEE Trans Cybern

Ke L, Zhang Q, Battiti R (2014) Hybridization of decomposition and local search for multiobjective optimization. IEEE Trans Cybern

Knowles J, Corne D (2000) M-PAES: a memetic algorithm for multiobjective optimization. In: 2000 Congress on evolutionary computation, vol 1, Piscataway, New Jersey. IEEE Service Center, pp 325–332

Knowles J, Thiele L, Zitzler E (2006) A tutorial on the performance assessment of stochastic multiobjective optimizers. 214, computer engineering and networks laboratory (TIK), ETH Zurich (revised version)

Konstantinidis A, Yang K, Zhang Q, Zeinalipour-Yazti D (2010) A multi-objective evolutionary algorithm for the deployment and power assignment problem in wireless sensor networks. Comput Netw 54(6):960–976

Krasnogor N, Gustafson S (2004) A study on the use of "self-generation" in memetic algorithms. Nat Comput 3(1):53–76

Krasnogor N, Smith J (2005) A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Trans Evol Comput 9(5):474–488

Larrañaga Pedro, Kuijpers Cindy M H, Murga Roberto H, Inza Iñaki, Dizdarevic S (1999) Genetic algorithms for the travelling salesman problem: a review of representations and operators. Artif Intell Rev 13(2):129–170

Li K, Zhang Q, Kwong S, Li M, Wang R (2014) Stable matching based selection in evolutionary multiobjective optimization. IEEE Trans Evol Comput 18:909–923

Liang Y-C, Lo M-H (2010) Multi-objective redundancy allocation optimization using a variable neighborhood search algorithm. J Heuristics 16(3):511–535

Li Y, Cai X, Fan Z, Zhang Q (2014) An external archive guided multiobjective evolutionary approach based on decomposition for continuous optimization. In: IEEE world congress on computational intelligence

Li Hui, Landa-Silva D (2011) An adaptive evolutionary multi-objective approach based on simulated annealing. Evol Comput 19:561–595

Li H, Zhang Q (2009) Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II. IEEE Trans Evol Comput 13(2):284–302

Lust T, Jaszkiewicz A (2010) Speed-up techniques for solving large-scale biobjective tsp. Comput OR 37(3):521–533

Lust T, Teghem J (2010) Two-phase pareto local search for the biobjective traveling salesman problem. J Heuristics 16(3):475–510

Mart Rl (2003) Multi-start methods. Int Ser Oper Res Manag Sci 57:355–368

Maulik U, Saha S, Deb K (2008) A simulated annealing-based multiobjective optimization algorithm: amosa. IEEE Trans Evol Comput 12(3):269–283

Miettinen K (1999) Nonlinear multiobjective optimization. Kluwer Academic Publishers, Boston

Mining challenges 2007 and 2009 of ieee working conf. mining software repositories (msr). Technical report

Nguyen QH, Ong Y-S, Krasnogor N (2007) A study on the design issues of memetic algorithm. In: IEEE congress on evolutionary computation, pp 2390–2397

Nguyen QH, Ong Y-S, Lim M-H (2009) A probabilistic memetic framework. IEEE Trans Evol Comput 13(3):604–623

Nguyen QH, Ong Y-S, Krasnogor N (2009) Adaptive cellular memetic algorithm. Evol Comput 17(2):231–256

Ong Y-S, Lim M-H, Zhu N, Wong KW (2006) Classification of adaptive memetic algorithms: a comparative study. IEEE Trans Syst Man Cybern Part B 36(1):141–152

Ong Y-S, Lim M-H, Chen X (2010) Memetic computation—past, present & future [research frontier]. IEEE Comput Intell Mag 5(2):24–31

Ong Yew-Soon, Keane Andy J (2004) Meta-lamarckian learning in memetic algorithms. IEEE Trans Evol Comput 8(2):99–110

Papadimitriou CH, Steiglitz K (1998) Combinatorial optimization: algorithms and complexity, Dover

Paquete L, Sttzle T (2009) Design and analysis of stochastic local search for the multiobjective traveling salesman problem. Comput Oper Res 36(9):2619C2631

Peng W, Zhang Q, Li H (2009) Comparison between MOEA/D and NSGA-II on the multi-objective travelling salesman problem. In: Multi-objective memetic algorithms, vol 171. Springer, Berlin, pp 309–324

Peter A, Bosman N (2012) On gradients and hybrid evolutionary algorithms for real-valued multiobjective optimization. IEEE Trans Evol Comput 16(1):51–69

Rodriguez-Alvarez M, Rojas I, Rojas F, Puntonet CG, Martin R (2004) Blind source separation in post-nonlinear mixtures using competitive learning, simulated annealing, and a genetic algorithm. IEEE Trans Syst Comput 34(4):407–416

Sato H, Aguirre HE, Tanaka K (2007) Local dominance and local recombination in MOEAs on 0/1 multiobjective knapsack problems. Eur J Oper Res 181(3):1708–1723

Shim VA, Tan KC, Cheong CY (2012) A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem. IEEE Trans Syst Man Cybern Part C 42(5):682–691

ShimVA, Tan KC, Tang H (2014) Adaptive memetic computing for evolutionary multiobjective optmization. IEEE Trans Cybern

Sindhya K, Miettinen K, Deb K (2013) A hybrid framework for evolutionary multi-objective optimization. IEEE Trans Evol Comput 17(4):495–511

Sudholt D (2006) Local search in evolutionary algorithms: the impact of the local search frequency. In: 17th international symposium on algorithms and computation (ISAAC)

Tan KC, Chew YH, Lee LH (2006) A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. Comput Optim Appl 34(1):115–151

Ulungu EL, Teghem J, Fortemps Ph, Tuyttens D (1999) MOSA method: a tool for solving multiobjective combinatorial optimization problems. J Multi Criteria Decis Anal 8(4):221–236

Wei P, Qingfu Z (2012) Network Topology planning using MOEA/D with objective-guided operators. In: Parallel problem solving from nature—PPSN XII, vol 7492, pp 62–71

Xuan J, Jiang H, Ren Z, Luo Z (2012) Solving the large scale next release problem with a backbone-based multilevel algorithm. IEEE Trans Softw Eng 38(5):1195–1212

Zhang Y, Harman M, Mansouri SA (2007) The multi-objective next release problem. In: Thierens D (ed) 2007 genetic and evolutionary computation conference (GECCO'2007), vol 1, London. ACM Press, pp 1129–1136

Zhang Q, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731

Zhang Q, Liu W, Li H (2009) The performance of a new version of moea/d on cec09 unconstrained mop test instances. Working Report CES-491, School of CS and EE, University of Essex

Zhang Y, Rockett PI (2009) A generic multi-dimensional feature extraction method using multiobjective genetic programming. Evol Comput 17(1):89–115

Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans Evol Comput 3(4):257–271