A Probabilistic Pareto Local Search based on Historical Success Counting for Multiobjective Optimization

Xinye Cai,Xin Cheng
Computer Science and Technology
Nanjing University of Aeronautics and Astronautics
Nanjing,Jiangsu, P. R. China
xinye@nuaa.edu.cn

Zhun Fan
Department of Electrical Engineering
Shantou University
Shantou, Guangdong, P. R. China
zfan@stu.edu.cn

ABSTRACT

In this paper, we propose a multiobjective probabilistic Pareto local search to address combinatorial optimization problems (COPs). The probability is determined by the success counts of local search offspring entering an external domination archive and this probabilistic information is used to further guide the selection of promising solutions for Pareto local search. In addition, simulated annealing is integrated in this framework as the local refinement process. This multiobjective probabilistic Pareto local search algorithm (MOPPLS), is tested on two famous COPs and compared with some well-known multiobjective evolutionary algorithms. Experimental results suggest that MOPPLS outperforms other compared algorithms.

Categories and Subject Descriptors

D.2.0 [**Computing Methodologies, Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

Keywords

Multiobjective Optimization; Probabilistic; Parteo local search; Aggregation

1. INTRODUCTION

Combinatorial optimization problems(COPs) have been extensively studied by scientists for many years because of their wide application in real life. EAs is a class of stochastic search techniques that have already been successfully applied to a wide range of optimization problems. Originally, EAs draw the inspiration from evolution and mimic genetic operations such as crossover and mutation to sample, learn and adapt from a population of candidate solutions. However, it takes them relatively long time to locate the exact local optimum within the region of convergence when facing complex optimization problems [9]. Genetic local searches [14], also referred to in the literature as Memetic algorithms(MAs), Baldwinian EAs, Lamarckian EAs and cultural algorithms, are extensions of EAs with the introduction of individual learning as a separate process of local refinement for accelerating search.

In addition, many real-world combinatorial optimization problems also involve several conflicting objectives to be simultaneously optimized. Along with it, many local search techniques in single objective heuristics such as variable neighborhood search [11], guided local search [1], simulated annealing [15] and ant colony optimization [6] have been generalized to solve multiobjective COPs. Different from single objective optimization problem where a single optimal solution exists, a multiobjective optimization problem usually has a set of non-dominated solutions, which represent the trade-off between the multiple objectives. Over the past two decades, two different fitness assignment schemes for mutliobjective local search have been frequently used [10, 17], as follows.

- *Parteo local search*(*domination*) [12, 13]: It is considered as a natural extension of single objective local search methods. It explores some or all of the neighbors of a set of mutually non-dominated solutions to find new non-dominated solutions for updating at each generation. The replacement of non-dominated solutions is based on the concept of Pareto dominance.
- *Aggregation*(*decomposition*) [16]: A multiobjective optimization problem(MOP) is transformed into a single objective optimization problem by linearly or nonlinearly aggregating all the objectives into a single objective. Under mild conditions, an optimal solution to this single objective problem is Pareto-optimal to the MOP. Thus, a single objective local search method can be used for finding a set of Pareto optimal solutions by solving a set of such single objective problems with different weights.

Motivated by [10], we propose an multiobjective probabilistic Pareto local search based on aggregation in this paper. In this paper, we propose a multiobjective probabilistic Pareto local search to address combinatorial optimization problems (COPs). The probability is determined by the success counts of local search offspring entering an external archive and this probabilistic information is used to further guide the selection of promising solutions for Pareto local search.

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.
Copyright 2014 ACM 978-1-4503-2881-4/14/07 ...$15.00.
http://dx.doi.org/10.1145/2598394.2610011.

The main contributions of this paper are as follows.

- A probabilistic Pareto local search framework is proposed and the simulated annealing is integrated as the local search metaheuristic, to tackle multiobjective COPs.
- The proposed approach was tested on multiobjective software next release problem(MONPR) and multiobjective traveling salesman problem(MOTSP) on synthetic test instances and real test instances mined from bug repository [20]. For the real test instances of MONRP, our proposed multiobjective approaches are compared with the state-of-the-art single objective approaches as well as well-known MOEAs. The results show our approach outperforms all the other approaches in comparison.

The rest of this paper is organized as follows. Since this paper focuses on addressing multiobjective optimization problems, Section 2 revisits basic concepts of multiobjective optimization(MOP). The following Section 3 is mainly dedicated to the detailed description of our proposed multiobjective probabilistic Pareto local search framework. Section 4 introduces two representative benchmark multiobjective COPs, that are multiobjective traveling salesman problem (MOTSP) and multiobjective next release problem (MONRP). Experimental studies and discussions are detailed in Section 5. The final conclusions of this paper is made in Section 6.

2. MOP AND MOEAS

Since this paper focuses on addressing multiobjective optimization, this section contributes to the introduction of multiobjective optimization problems and multiobjective evolutionary algorithms. In the multiobjective optimization problem(MOP), two or more usually conflicting objectives are required to be optimized simultaneously. A MOP can be defined as:

Given a problem involving $\boldsymbol{n}$ decision variables $x_1, x_2, \ldots, x_n$ in a search space $\boldsymbol{X} \subset \Re^n$, we assume, without loss of generality, $\boldsymbol{m}$ objectives $f_1(\cdot), \ldots, f_m(\cdot)$ in *objective function space* $\boldsymbol{Y} \subset \Re^m$, are to be minimized.

Minimize $\mathbf{f}(\vec{x}) = ((f_1(x_1, x_2, \ldots, x_n)), \ldots, f_m(x_1, x_2, \ldots, x_n))$

The vector function is a mapping $\boldsymbol{f}\colon \boldsymbol{X} \to \boldsymbol{Y}$.

In MOP, it is usually not possible to find a single solution which is optimal for all the objectives. Instead, many good solutions may exist. These solutions are always "trade-offs" or good compromises among the objectives. Since the conventional concept of optimality does not hold, a concept of Pareto optimality is adopted. The Pareto optimality concept, which was first proposed by Edgeworth and Pareto [18], is formally defined as follows [3]:

Let $\vec{x}$,$\vec{y}$ be two vectors of decision variables in MOP. $\vec{x}$ is considered to dominate $\vec{y}$(written as $\vec{x} \prec \vec{y}$) iff they satisfy the conditions:

$$\begin{array}{l} \vec{x}, \vec{y} \in \mathbf{X}, \forall i \in 1, \ldots, m | f_i(\vec{x}) \le f_i(\vec{y}) \\ \vec{x}, \vec{y} \in \mathbf{X}, \exists j \in 1, \ldots, m | f_j(\vec{x}) < f_j(\vec{y}) \end{array} \tag{1}$$

On the contrary, a decision vector $\vec{x}$ is considered to be a non-dominated solution iff there is no other solution that satisfies eq.(1). The set of all non-dominated solutions forms a *Pareto optimal set*:

The projection of the Pareto optimal set $\boldsymbol{P}$ in the $\boldsymbol{m}$ dimensional objective function space $\boldsymbol{Y}$ is called *Pareto front*, $\boldsymbol{F}$.

$$\boldsymbol{F} = \{(f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})) | \vec{x} \in \boldsymbol{P}\} \tag{2}$$

3. THE PROPOSED ALGORITHM

3.1 The Framework

For convenience, the proposed approach is named as multiobjective probabilistic Pareto local search (MOPPLS). MOPPLS decomposes the MOP into N single objective optimization subproblems. For simplicity, the weight sum approach is adopted in this paper, it requires N weight vectors:

$$\lambda^j = (\lambda_1^j, \ldots, \lambda_m^j) \;\; j = 1, \ldots, N. \tag{3}$$

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

$$\begin{array}{ll} \text{minimize} & g_k^{ws}(x) = \sum_{i=1}^m \lambda_i^j f_i(x) \\ \text{subject to} & x \in \Omega \end{array} \tag{4}$$

For each $k = 1, \ldots, N$, set $B(k)$ contains the indexes of the T closest weight vectors to λ^k in terms of the Euclidean distance. If $i \in B(k)$, then subproblem i is defined as a neighbor of subproblem k. At each generation, the proposed framework maintains several populations as follows. 1) Decomposition population $P = \{x^1, \ldots, x^N\}$, where x^k is the best solution found so far for subproblem k. 2) Domination population A, which contains N solutions selected by using the non-dominated sorting selection and crowding distance methods proposed in [3, 4]. 3) L, which contains the solutions generated by local search and Y, which contains the solutions generated by global search.

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

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

3.2 Initialization

A multiobjective optimization problem is originally decomposed into a number of subproblems based on **eq. 3**. A multiobjecitve COP is decomposed into corresponding N subproblems, when N uniformly distributed weight vectors $\boldsymbol{\lambda^1}, \ldots, \boldsymbol{\lambda^N}$ are generated. Meanwhile, a population P, each solution of which is the best solution found for each subproblem, is randomly generated and assigned to the external archive A, which is used to store diverse non-dominated solutions. The whole process of initialization is shown in Step 1 of **Algorithm 1**.

3.3 Local Search

The whole process of local search is shown in Step 2 of **Algorithm 1**. In this step, local search(SA in our case) is applied to the decomposition population P. Furthermore, we consider individual subset of the population that should undergo the refinement process, which is termed the issue of *the frequency of refinement and individual subset selection* in [19]. In this paper, we propose the adaptive mechanism to adaptively select individuals for local search in Step 2a. We also use two common strategies selecting solutions for local search as the baseline for comparisons. The total two

```
Algorithm 1:MOPLS
Input:
  1. Multiobjective COPs;
  2. a stopping criterion;
  3. $N$: the number of subproblems; the population size of $P$ and $A$;
  4. a uniform spread of $N$ weight vectors: $\boldsymbol{\lambda^1}, \ldots, \boldsymbol{\lambda^N}$;
  5. the size of the neighborhood of each subproblem, denoted as $T$;
Output: A set of non-dominated solutions $A$;
Step 1: Initialization:
  a) Decompose the original multiobjective COP into $N$ subproblems associated with $\boldsymbol{\lambda^1}, \ldots, \boldsymbol{\lambda^N}$.
  b) Generate an initial population $P = \{x^1, \ldots, x^N\}$ randomly.
  c) Assign $A = P$.
  d) Compute 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:Probabilistic Local search
  a) Select $N$ solutions from $P$.
  b) Perform local search to the selected solutions to generate $N * J$ solutions.
Step 3: Global search
  For each i $\in P$, do:
  a) Randomly select two indexes $k$ and $l$ from $B(i)$.
  b) Apply one point crossover and bit-wise mutation operators to $x_k$ and $x_l$ to generate $y_i$ for subproblem $i$.
Step 4: Population update
  /* Use $Y$ to update decomposition population $P$*/
  a) For each i $\in P$, do:
  b) If $y_i$ is generated from subproblem $i$, for each index $k \in B(i)$, if $g^{ws}(y_i|\lambda^k) \leq g^{ws}(x_k|\lambda^k)$, then set $x_k = y_i$.
  /* Use $L$ to update decomposition population */
  c) Set $j = 1$.
  d) 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 $x_k = L_j$.
  e) Set $j \to j + 1$. If $j \leq J * N$, go back to Step 4d.
  /* Use $Y$ and $L$ to update domination population $A$*/
  f) Merge $Y$ with $A$ and new solution set L in Local search;to obtain $Z = A \cup Y \cup L$; sort the merged population $Z$ with fast non-dominated sorting and crowding distance approach of NSGA-II [4] and the best $N$ solutions form the new $A$.
Step 5: Termination
  a) If stopping criteria are satisfied, terminate the algorithm and output $A$. Otherwise, go to Step 2.
```

strategies of the selections of individuals for local search are as follows.

1) *Global search:* For each generation, all the solutions in the decomposition population P are selected to undergo local refinement. The proposed approach that used this selection strategy for local search is called MOPLS in our paper.

2) *Probabilistic Pareto local search algorithm:* In this paper, we proposed an adaptive solution selection for local search based on the learning of domination archive. The domination archive contains very valuable global convergence and diversity information after applying Pareto dominance and diversity maintenance mechanism. Such valuable information can be used to guide the selection of solutions in the decomposition population for local search. In this paper, we adopt non-dominated sorting and crowding distance method in NSGA-II. More detailed description of this mechanism is as follows.

Step 1 Initialization: Initialize P and A.

Step 2 Local Search: Perform local search to P or A and generate a set of new solutions L.

Step 3 Global Search: Perform global search to generate a set of N solutions Y from P.

Step 4 Population Update: Use Y and L to update P and A.

Step 5 Stopping Condition: If a preset stopping condition is met, output A. Otherwise, go to Step 2.

Multiple new solutions are generated by the local search procedure. A new solution is called successful if it enters A. Note that whether or not a new solution enters A is determined by Pareto dominance sorting and crowding distance method proposed in [3,4]. Thus the number of successful solutions generated through local search procedures of a subproblem records the global convergence and diversity information of this subproblem, which can be used to guide the selection of subproblems for local search. We count $ds_{i,g}$ as the number of successful solutions through the local search of the ith subproblem at generation g. The total successful solutions within a certain fixed number of pervious generations, defined as the learning generations(LGs), is used to calculate the probability for each subproblem to be selected for local search as follows.

At each generation $G > LGs - 1$, the probability of choosing the ith$(i = 1, 2, \ldots, N)$ subproblem is calculated by

$$prob_{i,G} = \frac{D_{i,G}}{\sum_{i=1}^{N} D_{i,G}} \tag{5}$$

where

$$D_{i,G} = \frac{\sum_{g=G-LGs}^{G-1} ds_{i,g}}{\sum_{g=G-LGs}^{G-1} total_{i,g}} + \epsilon, (i = 1, 2, \ldots, N; G > LGs) \tag{6}$$

$D_{i,G}$ represents the proportion of successful solutions through the local search of the ith subproblem within the previous LGs. $ds_{i,g}$ is the number of non-dominated solutions through the local search of the ith subproblem and $total_{i,g}$ is the total number of non-dominated solutions through the local search of all subproblems within the previous LGs. A small constant value $\epsilon = 0.002$ is used to avoid the possible zero selection probabilities. We also normalize $D_{i,G}$ in order to make the summation of $D_{i,G}$ for all subproblems into 1.

A Roulette Wheel Selection is applied to select a subproblem based on the probabilities calculated by eq. 5. The selected subproblem(solution) undergoes local search metaheuristics. Each solution generates J solutions after applying local search. By repeating this procedure N times, a population L with $N * J$ solutions is generated by the local search procedure. To distinguish with our proposed MOPPLS, the algorithm that uses the ordinary Pareto local search is named as multiobjective Pareto local search (MOPLS).

3.4 Global Search

The whole process of the global search is presented in Step 3 of **Algorithm 1**. For the ith subproblem, two parents are selected from its T neighboring subproblems, as shown in Step 3a. In the following Step 3b, a simple one point crossover and a bit-wise mutation operators are applied to the parents to generate an offspring y_i. By repeating this procedure(Step 3a - 3b) N times, an offspring population $Y = \{y_1, \ldots, y_N\}$ is generated.

3.5 Population Update

In Step 4, the newly generated offspring population Y is used to update both P and A. For each solution y_j in Y, suppose it was generated from subproblem i. Step 4b considers all the neighbors of the i-th subproblem, it replaces all neighbors x_k with y_j if y_j performs better than x_k with regard to the k-th subproblem. The update of domination archive is presented in Step 4d. The offspring population Y is merged with A, and population L is produced from the local search in Step 2 of **Algorithm 1**, then the combined population Z(the combinations of Y, A and L) is sorted by the fast non-dominated sorting method and the crowding distance procedure. The best N solutions are kept for A.

4. BENCHMARK PROBLEMS

In this paper, we consider two NP-hard multiobjective COPs: the multiobjective software next release problem(an instance of knapsack problem) and the multiobjective traveling salesman problem. These combinatorial optimization problems have been widely used on testing the performance of various MOEAs [2,7,21].

4.1 The Single and Biobjective Next Release Problem

Assumes that an software system is associated with several *customers* whose requirements need to be considered in the next release. The set S of the customers is denoted by $s_i(1 \le i \le m))$. The set R of all the *requirements* that need to be considered is denoted by $r_j(1 \le j \le n))$.

Implementation of each requirement need to be allocated with certain number of resources, A cost vector C is represented by $c_j(1 \le j \le n))$, where c_i indicates that cost of the requirement $r_i \in R$. A request $q_{ij} \in Q$ represent whether a customer s_i request a requirement r_i, q_{ij}=1 denotes that s_i requests r_i or q_{ij}=0 denotes not. Moreover, we assume that customers can gain different profits, when their requests are satisfied. We represent these profits as W=$w_i(1 \le i \le m))$, where w_i denotes the profit when the requests of customer s_i are satisfied in the next release.

The goal of the *Single objective NRP* is to find an optimal solution $X^\star$, to maximize $Profit(X) = \sum_{(i,1)\in X} w_i$ gained by customers, which is the sum of profits gained by the selected customers, subject to $Cost(X) \le b$, where b is a predefined budget constraint and $cost(X) = \sum_{r_k \in R(X)} c_k$, where $R(X) = \bigcup_{(i,1)\in X, q_{ij}=1} \{r_j\}$ is the requirements for X. Similarly, For *the multiobjective NRP*, The total profit for a solution X is defined by $W(X) = \sum_{(i,1)\in X} s_i$,which is the sum of profits gained by the selected customers. The required $Cost$ for implementing the requirements of a solution X is $Cost(X) = \sum_{r_k \in R(X)} c_k$, where $R(X) = \bigcup_{(i,1)\in X, q_{ij}=1}$ $\{r_j\}$ is the requirements for X.

4.2 The Multiobjective Traveling Salesman Problems

The Traveling Salesman Problem (TSP) can be modeled 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 task is to find a Hamiltonian cycle of minimal length, which visits each vertex exactly once. In the case of the multiobjective TSP (MOTSP), each edge $e_{i,j}$ is associated with multiple values such as cost, length, traveling time, etc. Each of them corresponds to one criterion. Mathematically, the MOTSP can be formulated as:

$$minimize \quad f_i(\pi) = \sum_{j=1}^{n-1} c^{(i)}_{\pi(j),\pi(j+1)} + c^{(i)}_{\pi(1),\pi(n)} \qquad i = 1, ..., m \tag{7}$$

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

5. EXPERIMENTAL STUDIES AND DISCUSSIONS

In this section, we conduct experiments on both MONRP and MOTSP to validate our proposed MOPPLS. Both synthetic and real test instances are used and the introduction of them is included in this section, followed by the experimental setups.

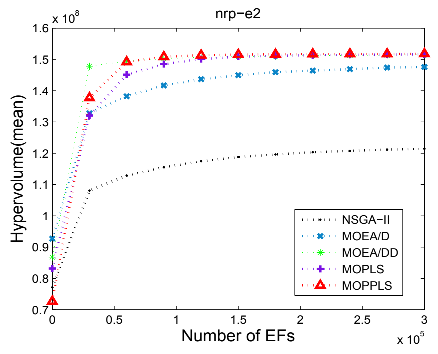
5.1 Test Instances and Experimental Setups

For both MONRP and MOTSP, randomly generated synthetic test instances were used. For MONRP, four sets of test instances were generated with various number of requirements and customers. Similarly, MOTSP test instances of different size were generated. For convenience, the test instances of MONRP and MOTSP are named in the following manners. For example, a 30-customers-and-300-requirements MONRP test instance is denoted as($cust30$ $req300$). A 500-city MOTSP is denoted as (tsp-$c500$). For those real NRP test instances were mined from bug repositories of two open source software projects, namely, Eclipse [5] and Gnome [8]. For convenience, test instances are denoted as nrp-e and nrp-g, respectively, in this paper.

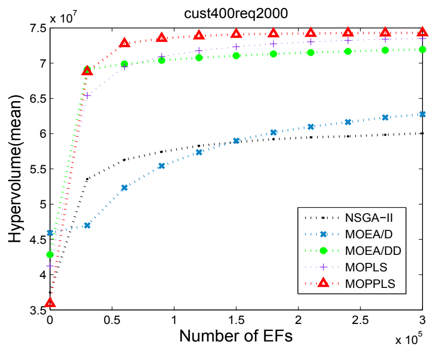
The parameter settings of the compared algorithms for MONRP and MOTSP as follows: the population size of all approaches is set to 200 for MONRP and 100 for MOTSP. The number of neighbors in our proposed MOPLS is set to 10 for them. We set sensitivity of parameter J, which is set to 5 in our experiments. For simulated annealing as the local search meta-heuristic, its initial temperature(Te) is 100 and scheduling factor(α) is set to 0.99. In addition, the number of function evaluations for MONRP is set to 300,000 and all the real test instances. In MOTSP, the number of function evaluation is set to 600,000.

5.2 The Comparison of Various Algorithms

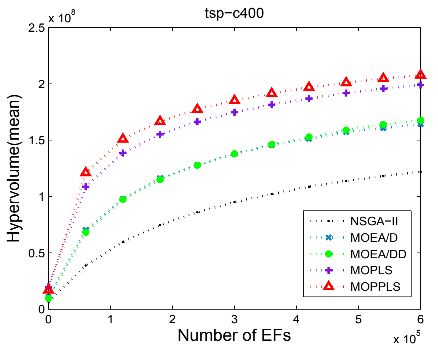
In this section, we conduct experiments to compare multiobjective Pareto local search(MOPLS) and probabilistic Pareto local search(MOPPLS) with other classical multiobjective optimization approaches. Fig. 1 shows the performances of five algorithms in terms of hypervolume values during the whole evolutionary process on two MONRP instances(both synthetic and real NRP Instances) and two MOTSP instances. These five approaches include MOPLS,



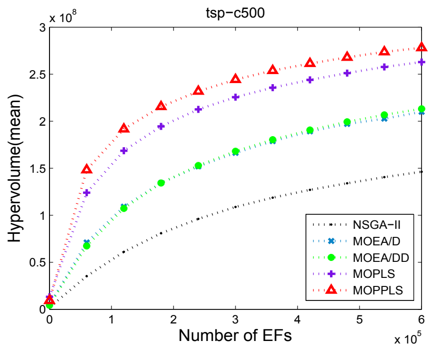
(a) nrp-e2



(b) cust400req2000



(c) tsp-c400



(d) tsp-c500

Figure 1: Convergence graphs in terms of I_H obtained by various algorithms on MONRP and MOTSP

MOPPLS, NSGA-II, MOEA/D and MOEA/DD. NSGAII and MOEA/D are well-known domination and decomposition based approach, respectively. MOEA/DD represents the proposed MOPPLS when no probabilistic Pareto local search is applied or MOPLS when no Pareto local search is applied. The convergence plots in Fig. 1 show that MOPPLS outperforms all other compared algorithms. To validate the results statistically, we use Table 1 to illustrate the mean and standard deviation of hypervolume(I_H), obtained by the five algorithms on MONRP instances and MOSTP instances. It can be observed in the Table 1 that the proposed MOPLS outperforms NSGA-II, MOEA/D and MOEA/DD in terms of hypervolume metric, on all test instances, and MOPPLS further outperforms MOPLS. In other words, MOPPLS always has the best performance compared with other approaches in all test instances.

6. CONCLUSION

In this paper, we propose a multiobjective probabilistic Pareto local search to address combinatorial optimization problems (COPs). The probability is determined by the success counts of local search offspring entering an external domination archive and this probabilistic information is used to further guide the selection of promising solutions for Pareto local search. In addition, simulated annealing is integrated in this framework as the local refinement process. This multiobjective probabilistic Pareto local search algorithm (MOPPLS), is tested on two famous COPs and compared with some well-known multiobjective evolution-

Table 1: Mean and standard deviation of I_H obtained by various algorithms on MONRP and MOTSP

——		NSGAII	MOEA/D	MOEA/DD	*MOPLS*	*MOPPLS*
cust200req1000	mean	1.6518e+07	1.7793e+07	1.8665e+07	1.8732e+07	**1.8743e+07**
	std	1.9904e+05	1.0193e+05	3.7785e+04	2.5803e+04	1.4609e+04
cust300req1500	mean	3.1568e+07	2.6528e+07	3.5676e+07	3.6332e+07	**3.7184e+07**
	std	4.3181e+05	4.0284e+06	3.4274e+05	6.3052e+05	4.2604e+05
cust400req2000	mean	6.0050e+07	6.2748e+07	7.1928e+07	7.3481e+07	**7.4301e+07**
	std	8.7525e+05	1.1012e+06	5.1543e+05	5.2123e+05	2.2610e+05
cust500req2500	mean	7.6671e+07	7.6933e+07	9.6964e+07	1.0136e+08	**1.0239e+08**
	std	1.4111e+06	2.5817e+06	1.0428e+06	4.3044e+05	2.9678e+05
nrp-e1	mean	1.1617e+08	1.4268e+08	1.4619e+08	1.4635e+08	**1.4655e+08**
	std	1.2917e+06	6.6765e+05	1.8761e+05	1.6063e+05	7.4332e+04
nrp-e2	mean	1.2142e+08	1.4758e+08	1.5142e+08	1.5158e+08	**1.5178e+08**
	std	1.3331e+06	8.2047e+06	1.8171e+05	1.3569e+05	6.8453e+04
nrp-g1	mean	9.4570e+07	1.1548e+08	1.1737e+08	1.1746e+08	**1.1749e+08**
	std	1.1625e+06	4.0086e+05	6.2844e+04	3.1842e+04	1.6110e+04
nrp-g2	mean	7.3694e+07	8.7304e+07	8.7821e+07	8.7826e+07	**8.7829e+07**
	std	7.3188e+05	5.5541e+04	2.0288e+04	1.5200e+04	1.6525e+04
tsp-c300	mean	7.8647e+07	1.0028e+08	1.0172e+08	1.1757e+08	**1.2200e+08**
	std	1.0299e+06	1.0677e+06	1.4351e+06	1.2675e+06	1.5036e+06
tsp-c400	mean	1.2177e+08	1.6419e+08	1.6759e+08	1.9890e+08	**2.0758e+08**
	std	2.2033e+06	4.1758e+06	1.9895e+06	1.8527e+06	2.3867e+06
tsp-c500	mean	1.4621e+08	2.1001e+08	2.1324e+08	2.6294e+08	**2.7813e+08**
	std	2.8420e+06	2.7892e+06	3.6784e+06	3.5585e+06	2.8940e+06
tsp-c600	mean	3.0337e+08	4.4358e+08	4.4479e+08	5.3350e+08	**5.5480e+08**
	std	5.7284e+06	5.9462e+06	5.0039e+06	3.8984e+06	3.7734e+06

ary algorithms. Experimental results suggest that MOPPLS outperforms other compared algorithms.

7. REFERENCES

[1] A. Alsheddy and E. Tsang. Guided pareto local search based frameworks for biobjective optimization. *n IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8, 2010.

[2] X. Cai, O. Wei, and Z. Huang. Evolutonary approches for multi-objective next release problem. *Computing and Informatics*, (4):847–875, 2012.

[3] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons, Inc., New York, NY, USA, 2001.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[5] Eclipse. http://www.eclipse.org/, July 2011.

[6] 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. *European Journal of Operational Research*, 180(1):116–148, July 2007.

[7] A. Gaspar-Cunha. A Multi-Objective Evolutionary Algorithm for Solving Traveling Salesman Problems: Application to the Design of Polymer Extruders. In B. Ribeiro, R. F. Albrecht, A. Dobnikar, D. W. Pearson, and N. C. Steele, editors, *Adaptive and Natural Computing Algorithms*, pages 189–193, Coimbra, Portugal, March 2005. Springer.

[8] Gnome. http://www.gnome.org/, July 2011.

[9] D. C. Grosan and A. Abraham. Hybrid evolutionary algorithms: Methodologies, architectures, and reviews. In *Hybrid Evolutionary Algorithms.* Springer, Berlin, Germany, 2007.

[10] L. Ke, Q. Zhang, and R. Battiti. A simple yet efficient multiobjective combinatorial optimization method using decompostion and Pareto local search. *IEEE Trans on Cybernetics*, accepted, 2014.

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

[12] T. Lust and A. Jaszkiewicz. Speed-up techniques for solving large-scale biobjective tsp. *Computers & OR*, 37(3):521–533, 2010.

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

[14] Q. H. Nguyen, Y.-S. Ong, and M.-H. Lim. A probabilistic memetic framework. *IEEE Trans. Evolutionary Computation*, 13(3):604–623, 2009.

[15] U. M. S. Saha and K. Deb. A simulated annealing-based multiobjective optimization algorithm: Amosa. *IEEE Transactions on Evolutionary Computation*, 12(3):269–283, 2008.

[16] 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 Transactions on Systems, Man, and Cybernetics, Part C*, 42(5):682–691, 2012.

[17] K. Sindhya, K. Miettinen, and K. Deb. A hybrid framework for evolutionary multi-objective optimization. *IEEE Trans. Evolutionary Computation*, 17(4):495–511, 2013.

[18] W. Stadler. A survey of multicriteria optimization or the vector maximum problem, part i: 1776ĺc1960. *Journal of Optimization Theory and Applications*, 29(1):1–52, 1979.

[19] D. Sudholt. Local search in evolutionary algorithms: The impact of the local search frequency. In *17th International Symposium on Algorithms and Computation(ISAAC)*, 2006.

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

[21] Y. Zhang, M. Harman, and S. A. Mansouri. The Multi-Objective Next Release Problem. In D. Thierens, editor, *2007 Genetic and Evolutionary Computation Conference (GECCO'2007)*, volume 1, pages 1129–1136, London, UK, July 2007. ACM Press.