Comparison between MOEA/D and NSGA-II on the Multiobjective Travelling Salesman Problem

Wei Peng*
School of Computer Science and Technology,
National University of Defense Technology,
Changsha, Hunan, 410073, China
Email: wpeng@nudt.edu.cn

Qingfu Zhang
Department of Computing and Electronic Systems,
University of Essex,
Colchester, Essex, CO4 3SQ, UK
Email: qzhang@essex.ac.uk

Hui Li
School of Computer Science and IT,
University of Nottingham,
Jubilee Campus, Wollaton Road,
Nottingham, NG8 1BB, UK

Technical Report CES-478

Department of Computing and Electronic Systems
University of Essex

December 4, 2007

*This work was done during the author visited the University of Essex as an academic visitor.

Abstract

Most multiobjective evolutionary algorithms are based on Pareto dominance for measuring the quality of solutions during their search, among them NSGA-II is well-known. A very few algorithms are based on decomposition and implicitly or explicitly try to optimize aggregations of the objectives. MOEA/D is a very recent such an algorithm. One of the major advantages of MOEA/D is that it is very easy to use well-developed single optimization local search within it. This paper compares the performance of MOEA/D and NSGA-II on the multiobjective travelling salesman problem and studies the effect of local search on the performance of MOEA/D.

1 Introduction

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

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

where Ω is the decision (variable) space, R^m is the objective space, and $F : \Omega \rightarrow R^m$ consists of m real-valued objective functions.

Very often, no single solution can optimize all the objectives in a MOP since its objectives conflict each other. Pareto optimal solutions, which characterize optimal trade-offs among these objectives, are of practical interest in many real-life applications. A solution is called Pareto optimal if any improvement in one single objective must lead to deterioration in at least one other objective. The set of all the Pareto optimal solutions in the objective space is called the Pareto front (PF). Many MOPs may have a huge (or even infinite) number of Parto optimal solutions. It is very time-consuming, if possible, to obtain the complete PF. Most multiobjective evolutionary algorithms (MOEAs) are to find a set of representative Pareto solutions to approximate the whole PF.

The majority of existing MOEAs are based on Pareto dominance [1]-[3]. In these algorithms, the utility of each individual solution is mainly determined by its Pareto dominance relations with other solutions visited in the previous search. Since using Pareto dominance alone could discourage the diversity of search, some techniques such as fitness sharing and crowding have often been used as compensation in these MOEAs. Arguably, NSGA-II [4] is the most popular Pareto dominance based MOEAs. The characteristic feature of NSGA-II is its fast non-dominated sorting procedure for ranking solutions in its selection.

A Pareto optimal solution to a MOP could be an optimal solution of a scalar optimization problem in which the objective is an aggregation function of all the individual objectives. Therefore, approximation of the Pareto front can be decomposed into a number of scalar objective optimization subproblems. This is a basic idea behind many traditional mathematical programming methods for approximating the PF. A very small number of MOEAs adopt this idea to some extent, among them the Multiobjective Evolutionary Algorithm Based on Decomposition (MOEA/D) is a very recent one [5]. MOEA/D attempts to optimize these subproblems simultaneously. The neighborhood relations among these subproblems are defined based on the distances between their aggregation coefficient vectors. Each subproblem (i.e., scalar aggregation function) is

optimized in MOEA/D by using information only from its neighboring subproblems. One of the major advantages of MOEA/D over Pareto dominance based MOEAs is that single objective local search techniques can be readily used in MOEA/D.

We believe that comparison studies between MOEAs based on Pareto dominance and those using decomposition on different multiobjective optimization problems could be very useful for understanding the strengths and weaknesses of these different methodologies and thus identifying important issues which should be addressed in MOEAs.

This paper proposes an implementation of MOEA/D for the multiobjective travelling salesman problem and compares it with NSGA-II. The effect of local search on the performance of MOEA/D has also been experimentally studied. The paper is organized as follows. Section 2 presents the multiobjective travelling salesman problem and the local search method used in our experiments. Section 3 presents MOEA/D and NSGA-II. Section 4 gives the experimental setting and performance metrics. Section 5 presents the experimental results. Finally, section 6 concludes the paper.

2 Multiobjective Travelling Salesman Problem

2.1 Problem

Given a number of cities and the cost of travel between each pair of them, the travelling salesman problem is to find the cheapest tour of visiting each city exactly once and returning to the starting point. The single objective TSP is NP-hard [6].

Mathematically, in the multiobjective TSP (mo-TSP), the decision space Ω is the set of all the permutations of $1, 2, \ldots, n$, and the m objectives to minimize are:

$$\begin{aligned} f_1(x_1, \ldots, x_n) &= \sum_{i=1}^{n-1} d^1_{x_i, x_{i+1}} + d^1_{n,1} \\ &\vdots = \vdots \\ f_m(x_1, \ldots, x_n) &= \sum_{i=1}^{n-1} d^m_{x_i, x_{i+1}} + d^m_{n,1} \end{aligned} \tag{2}$$

where $x = (x_1, \ldots, x_n)$ is a permutation vector, and $d^k_{i,j}$ can be regarded as the travel cost from city i to j in the k-th objective.

The mo-TSP has been used as a benchmark problem in studying the performance of MOEAs in recent years [7]-[9].

2.2 2-opt Local Search for the TSP

The 2-opt local search is a very popular and efficient local search method for improving a solution in the single objective TSP problem. A 2-interchange move on a solution (tour) to the TSP is to break it into two paths by deleting two edges and reconnect them in the other possible way. The neighborhood of a solution x consists of all the solutions which can be obtained by applying a 2-interchange

move on it. In order to improve a solution x, 2-opt local search searches the neighborhood of x to find a solution y which is better than x and then replace x by y. This process is repeated until no solution in the neighborhood of x is better than y or a predefined stopping condition is met.

Comparisons of the costs of two neighboring solutions are major computational overhead in 2-opt local search. Since 2 neighboring solutions are different only in 2 edges, the cost difference of these two solutions can be computed with several basic operations. Therefore, the computational overhead of 2-opt local search is often not high.

One of the key issues in 2-opt local search is how to search the neighborhood. In our implementation, we adopt the first improvement strategy, i.e. search the neighboring solutions in a predetermined order and y is the first solution found which is better than x. To limit the computational cost, the number of neighborhoods the 2-opt local search searches is not allowed to exceed a predefined limit, L_s.

3 MOEA/D

3.1 Weighted Sum Approach

MOEA/D requires a decomposition approach for converting approximation of the PF of (1) into a number of single objective optimization problems. In principle, any decomposition approach can serve for this purpose, In the paper, we use the weighted sum approach [10]. This approach considers a convex combination of the different objectives. Let $\lambda = (\lambda_1, \ldots, \lambda_m)^T$ be a weight vector, i.e. $\lambda_i \geq 0$ for all $i = 1, \ldots, m$ and $\sum_{i=1}^{m} \lambda_i = 1$. Then the optimal solution to the following scalar optimization problem

$$\begin{array}{ll} \text{minimize} & g(x|\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x) \\ \text{subject to} & x \in \Omega \end{array} \tag{3}$$

is a Pareto optimal solution to (1), where we use $g(x|\lambda)$ to emphasize that λ is a coefficient vector in this objective function while x is the variables to be optimized. To generate a set of different Pareto optimal vectors, one can use different weight vectors λ in the above scalar optimization problem. If the PF is convex (concave in the case of maximization), this approach could work well. However, not every Pareto optimal vector can be obtained by this approach in the case of nonconvex PFs. To overcome these shortcomings, some effort has been made to incorporate other techniques such as ε-constraint into this approach, more details can be found in [10].

3.2 General Framework

MOEA/D needs to decompose the MOP under consideration. Any decomposition approaches can serve for this purpose. In the following description, we suppose that the weighted sum approach is employed. It is very trivial to modify the following MOEA/D when other decomposition methods are used.

Let $\lambda^1, \ldots, \lambda^N$ be a set of even spread weight vectors. The problem of approximation of the PF of (1) can be decomposed into N scalar optimization

subproblems by using the weighted sum approach and the objective function of the j-th subproblem is:

$$g(x|\lambda^j) = \sum_{i=1}^{m} \lambda_i^j f_i(x) \tag{4}$$

where $\lambda^j = (\lambda_1^j, \ldots, \lambda_m^j)^T$. MOEA/D minimizes all these N objective functions simultaneously in a single run.

Note that g is continuous of λ, the optimal solution of $g(x|\lambda^i)$ should be close to that of $g(x|\lambda^j)$ if λ^i and λ^j are close to each other. Therefore, any information about these g's with weight vectors close to λ^i should be helpful for optimizing $g(x|\lambda^i)$. This is a major motivation behind MOEA/D.

In MOEA/D, a neighborhood of weight vector λ^i is defined as a set of its several closest weight vectors in $\{\lambda^1, \ldots, \lambda^N\}$. The neighborhood of the i-th subproblem consists of all the subproblems with the weight vectors from the neighborhood of λ^i. Each subproblem has its best solution found so far in the population. Only the current solutions to its neighboring subproblems are exploited for optimizing a subproblem in MOEA/D.

At each generation t, MOEA/D with the weighted sum approach maintains:

- a population of N points $x^1, \ldots, x^N \in \Omega$, where x^i is the current solution to the i-th subproblem;
- $FV^1, \ldots, FV^N$, where FV^i is the F-value of x^i, i.e., $FV^i = F(x^i)$ for each $i = 1, \ldots, N$;
- an external population EP, which is used to store nondominated solutions found during the search.

The algorithm works as follows:

Input: • MOP (1);
- a stopping criterion;
- N : the number of the subproblems considered in MOEA/D;
- a uniform spread of N weight vectors: $\lambda^1, \ldots, \lambda^N$;
- T : the number of the weight vectors in the neighborhood of each weight vector.

Output: EP.

Step 1 Initialization

Step 1.1 Set $EP = \emptyset$.

Step 1.2 Compute the Euclidean distances between any two weight vectors and then work out the 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 λ^i.

Step 1.3 Generate an initial population $x^1, \ldots, x^N$ randomly or by a problem-specific method. Set $FV^i = F(x^i)$.

Step 2 Update

For $i = 1, \ldots, N$, do

Step 2.1 Reproduction: Randomly select two indexes k, l from $B(i)$, and then generate a new solution y from x^k and x^l by using genetic operators.
Step 2.2 Improvement: Apply a problem-specific repair/improvement heuristic on y to produce y'.
Step 2.3 Update of Neighboring Solutions: For each index $j \in B(i)$, if $g(y'|\lambda^j) \leq g(x^j|\lambda^j)$, then set $x^j = y'$ and $FV^j = F(y')$.
Step 2.4 Update of EP:
Remove from EP all the vectors dominated by $F(y')$.
Add $F(y')$ to EP if no vectors in EP dominate $F(y')$.

Step 3 Stopping Criteria If stopping criteria is satisfied, then stop and output EP. Otherwise go to **Step 2**.

In initialization, $B(i)$ contains the indices of the T closest vectors of λ^i. We use the Euclidean distance to measure the closeness between any two weight vectors. Therefore, λ^i's closest vector is itself and then $i \in B(i)$. If $j \in B(i)$, the j-th subproblem can be regarded as a neighbor of the i-th subproblem.

In the i-th pass of the loop in Step 2, the T neighboring subproblems of the i-th subproblem are considered. Since x^k and x^l in Step 2.1 are the current best solutions to neighbors of the i-th subproblem, their offspring y should hopefully be a good solution to the i-th subproblem. In Step 2.2, a problem-specific heuristic is used to repair y in the case when y invalidates any constraints, and/or optimize the i-th g. Therefore, the resultant solution y' is feasible and very likely to have a lower function value for the neighbors of i-th subproblem. Step 2.3 considers all the neighbors of the i-th subproblem, it replaces x^j with y' if y' performs better than x^j with regard to the j-th subproblem. FV^j is needed in computing the value of $g^(x^j|\lambda^j)$ in Step 2.3. Step 2.4 updates the external population.

Step 2.2 allows MOEA/D to be able to make use of a scalar optimization method very naturally. One can take the $g(x|\lambda^i)$ as the objective function in the heuristic in Step 2.2. Although it is one of the major features of MOEA/D, Step 2.2 is not a must in MOEA/D, particularly if Step 2.1 can produce a feasible solution.

3.3 NSGA-II

NSGA-II [4] is a very popular MOEA based on Pareto domination. NSGA-II maintains a population P_t of size N at generation t and generate P_{t+1} from P_t in the following way.

Step 1 Use selection, crossover and mutation to create an offspring population Q_t from P_t.

Step 2 Choose N best solutions from $P_t \cup Q_t$ to form P_{t+1}.

The characteristic feature of NSGA-II is that it uses a fast nondominated sorting and crowding-distance estimation procedure for comparing qualities of different solutions in **Step 2** and selection in **Step 1**. The computational complexity of each generation in NSGA-II is $O(mN^2)$, where m is the number of the objectives and N is its population size.

Table 1: Experimental parameter Settings.

Genetic operator in three algorithms	inver-over crossover [8],
δ(Random inverse rate in inver-over crossover)	0.02,
Number of runs for each instances	10,
n (The number of cities)	500,1000,1500,
N (The population size in all the algorithms)	100,
Weight vectors in MOEA/D	100 uniformly distributed vectors
T in MOEA/D	25,
L_s in local Search	100,
CPU time used in each run (in second)	600 for the instance with 500 cities, 1600 for 1000 cites, and 4000 seconds for 1,500 cities.

To have a fair comparison, an external population is used in our implementation of NSGA-II to record all the non-dominated solutions found in the search.

4 Experiment settings

4.1 Multiobjective TSP instances

In this paper, we only consider two objectives. To generate a bi-objective test instance with n cities, a random distribution of cities in a $[0, n] \times [0, n]$ area is generated for each objective. The travel cost between any two cities is set to be the distance between them. We have tested three different instances in which the numbers of cities are 500, 1000 and 1500.

4.2 Parameter settings

In our experiments, we tested three algorithms, namely, MOEA/D without local search, MOEA/D with local search, and NSGA-II. The experiments were carried out in a desktop PC with Intel Pentium 4 CPU 3.20GHz and 1.50GB RAM. Table 1 gives the parameter settings used in the experiments.

4.3 Performance metrics

Due to the nature of multiobjective optimization, no single performance metric is always able to compare the performances of different algorithms properly. In our experiments, we use the following two metrics:

- **Set Coverage (C-metric)**: Let A and B be two approximations to the PF of a MOP, $C(A, B)$ is defined as the percentage of the solutions in B that are dominated by at least one solution in A, i.e.

$$C(A, B) = \frac{|\{u \in B | \exists v \in A : v \; dominates \; u\}\|}{|B|} \tag{5}$$

$C(A, B)$ is not necessarily equal to $1 - C(B, A)$. $C(A, B) = 1$ means that all solutions in B are dominated by some solutions in A, and $C(A, B) = 0$ means that no solution in B is dominated by a solution in A.

- **Distance from Representatives in the PF(D-metric)**: Let P^* be a set of uniformly distributed points along the PF. Let A be an approximation to the PF, the average distance from A to P^* is defined as: [5]

$$D(A, P^*) = \frac{\sum_{v \in P^*} d(v, A)}{|P^*|} \tag{6}$$

where $d(v, A)$ is the minimum Euclidean distance between v and the points in A. If $|P^*|$ is large enough to represent the PF very well, $D(A, P^*)$ could measure both the diversity and convergence of A in a sense. To have a low value of $D(A, P^*)$, set A must be very close to the PF and cannot miss any part of the whole PF.

As we do not know the actual PFs of the test instances, we use an approximation of the PF as P^*. The approximation of PF is obtained from all non-dominated solutions found in all the runs of the three algorithms.

5 Experiment Results

5.1 Comparison of MOEA/D and NSGA-II

We firstly compare the performance of MOEA/D algorithm without local search and the NSGA-II algorithm. Note that both algorithms use the same genetic operator.

5.1.1 C-metric

Table 2 presents the average C-metrics of MOEA/D and NSGA-II on different test instances. It is evident from this table that no solution obtained in NSGA-II dominates any solutions in MOEA/D at these selected observation times. Most solutions in NSGA-II are dominated by the solutions in MOEA/D in the test instance with 500 cities and all the solutions in NSGA-II are dominated by those in MOEA/D in the other two larger test instances while $n = 1000$ and 1500. A very interesting observation is that the C-metric of MOEA/D versus NSGA-II decreases as the execution time increases in the instance with 500 cities. But it does not happen in the two other larger instances. This implies that MOEA/D, compared with NSGA-II, is very promising in dealing with large scale problems.

In some real-world applications, the number of function evaluations matters. Table 3 gives the C-metrics of these two algorithms versus the number of function evaluations. Clearly, MOEA/D outperforms NSGA-II after a number of function evaluations in all the three instances.

5.1.2 D-metric

Figures 1-3 present the evolution of the D-metrics of two algorithms with the execution time. It is evident from these figures that MOEA/D always outperforms NSGA-II in terms of D-metric if two algorithms use the same execution time.

Table 2: C-metric vs execution time.

The instances	exec time (seconds)	NSGA-II vs MOEA/D	MOEA/D vs NSGA-II
500 cities	100	0.0	0.954062
	200	0.0	0.942587
	300	0.0	0.881377
	400	0.0	0.852883
	500	0.0	0.844775
	600	0.0	0.829717
1000 cities	200	0.0	0.997921
	400,600,...,1600	0.0	1.0
1500 cities	500,1000,...,4000	0.0	1.0

Table 3: C-metric vs. the number of function evaluations.

The instances	number of function evaluations	NSGA-II vs MOEA/D	MOEA/D vs NSGA-II
500 cities	0.5×10^6	0.968367	0.011017
	1.0×10^6	0.898114	0.031785
	1.5×10^6	0.761665	0.117353
	2.0×10^6	0.220869	0.402011
	2.5×10^6	0.009831	0.569514
	3.0×10^6	0.0	0.624579
1000 cities	0.5×10^6	0.511688	0.252936
	1.0×10^6	0.009333	0.614016
	1.5×10^6	0.0	0.745755
	2.0×10^6	0.0	0.810331
	2.5×10^6	0.0	0.857586
	3.0×10^6	0.0	0.903073
	3.5×10^6	0.0	0.927789
	4.0×10^6	0.0	0.947508
	4.5×10^6	0.0	0.961898
	5.0×10^6	0.0	0.979815
1500 cities	1.0×10^6	0.0	0.826044
	2.0×10^6	0.0	0.945171
	3.0×10^6	0.0	0.973439
	4.0×10^6	0.0	0.988983
	5.0×10^6	0.0	0.997826
	6.0×10^6	0.0	1.0
	7.0×10^6	0.0	1.0
	8.0×10^6	0.0	1.0

Figures 4-6 show how the D-metrics of two algorithms evolve with the number of function evaluations. In the instances with 1000 cities and 1500 cities, it is obvious that MOEA/D outperforms NSGA-II after a number of function evaluations in terms of D-metric. In the instance with 500 cities, D-metric in NSGA-II is slightly lower than that in MOEA/D. The reason is that the solutions generated in NSGA-II have better spread than those in MOEA/D in this instance. However, the final solutions found by MOEA/D are much closer to the real Pareto front than those in NSGA-II as shown in terms of C-metric.

These figures also indicate that MOEA/D can do more function evaluations than NSGA-II within the same execution time. This observation confirms the analysis of the computational complexity of these two algorithms in [5].

5.2 The Role of Local Search in MOEA/D

It is well known that hybridization of local search into evolutionary algorithms could improve the algorithmic performance very effectively. It is not easy to combine single optimization local search with Pareto domination based MOEAs such as NSGA-II. In contrast, MOEA/D can readily take the advantage of well-developed single objective optimization methods. To study if local search (Step 2.2 in MOEA/D) could enhance the algorithm, we have implemented MOEA/D with local search and compared it with MOEA/D without local search. Figures 7-9 compare the evolution of the D-metrics of MOEA/D with and without local search. It is clear that MOEA/D with local search significantly outperforms MOEA/D without local search, which implies local search could greatly enhance the algorithmic performance.

5.3 The Pareto Fronts found by Three Algorithms

To visualize the performance of the three algorithms, we plot the distribution of the final solutions in the objective space found by a random run in each algorithm on each test instance in figures 10-12. The results shown in these figures are consistent with the C-metrics and D-metrics. The differences among these three algorithms can be easily noticed from these figures.

6 Summary

Most multiobjective evolutionary algorithms such as NSGA-II are based on Pareto dominance. Single optimization local search is not easy to be combined with these algorithms. A very small number of multiobjective population algorithm are based on decomposition. We believe that comparison between these two different strategies could be very helpful for understanding their strengths and weaknesses and developing effective and efficient MOEAs. In this paper, we compared two MOEAs, where one is NSGA-II, a very popular MOEA based on Pareto domination, and the other is MOEA/D, a very recent MOEA based on decomposition. We have taken the multiobjective TSP as a test problem. Our experimental results have shown that MOEA/D without local search outperforms NSGA-II on the three test instances with the same execution time. We have also demonstrated that MOEA/D with local search works much better

than MOEA/D without local search. In the future, we will study the performance of MOEA/D on other optimization problems.

References

[1] Deb K (2001) Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, Baffins Lane, Chichester

[2] Coello C A, Veldhuizen D A, and Lamont G B (2002) Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer, Norwell, MA

[3] Tan K C, Khor E F, and Lee T H (2005) Multiobjective Evolutionary Algorithms and Applications. Springer-Verlag, New York

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

[5] Zhang Q and Li H (2007) MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Trans. on Evolutionary Computation 11: 712–731

[6] Johnson D S and McGeoch L A (1997) The traveling salesman problem: a case study. In: Aarts E and Lenstra J K (eds) Local Search in Combinatorial Optimization. John Wiley & Sons Ltd. 215–310

[7] Paquete L and Stützle T (2003) A Two-Phase Local Search for the Biobjective Traveling Salesman Problem. In: Fonseca C M, Fleming P J, Zitzler E, Deb K, and Thiele L (eds) Evolutionary Multi-Criterion Optimization (EMO'2003). Springer. 479–493

[8] Yan Z, Zhang L, Kang L and Lin G (2003) A New MOEA for Multiobjective TSP and Its Convergence Property Analysis. In: Fonseca C M, Fleming P J, Zitzler E, Deb K, and Thiele L (eds) Evolutionary Multi-Criterion Optimization (EMO'2003). Springer. 342–354

[9] 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. European Journal of Operational Research 180: 116–148

[10] Miettinen K (1999) Nonlinear Multiobjective Optimization. Kluwer, Norwell, MA

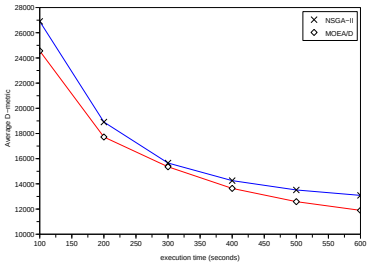

Figure 1: D-metric vs. execution time in the instance with 500 cities.

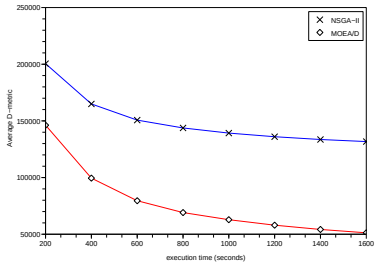

Figure 2: D-metric vs. execution time in the instance with 1000 cities.

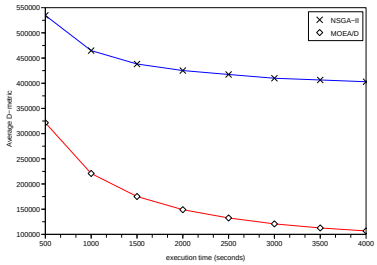

Figure 3: D-metric vs. execution time in the instance with 1500 cities.

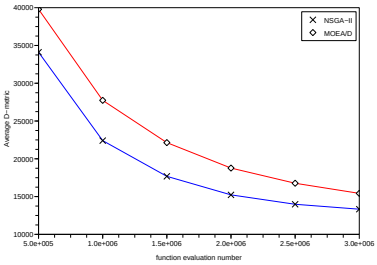

Figure 4: D-metric vs. the number of function evaluations in the instance with 500 cities.

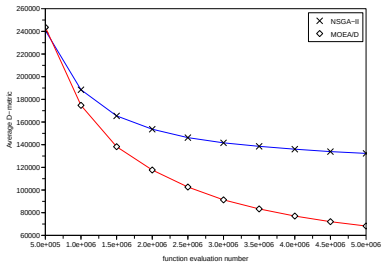

Figure 5: D-metric vs. the number of function evaluations in the instance with 1000 cities.

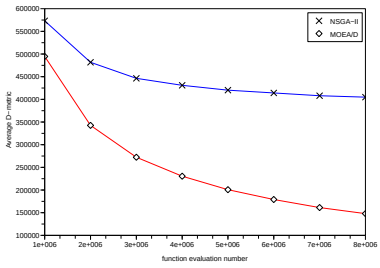

Figure 6: D-metric vs. the number of function evaluations in the instance with 1500 cities.

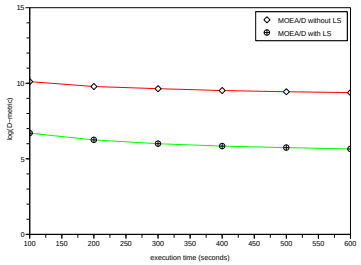

Figure 7: D-metric vs. execution time in MOEA/D with and without local search for the instance with 500 cities. The scale of the D-metric is log.

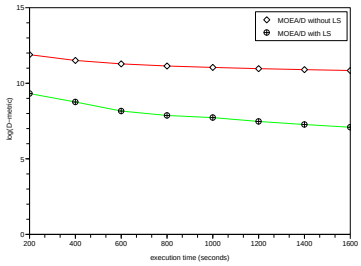

Figure 8: D-metric vs. execution time in MOEA/D with and without local search for the instance with 1000 cities. The scale of the D-metric is log.

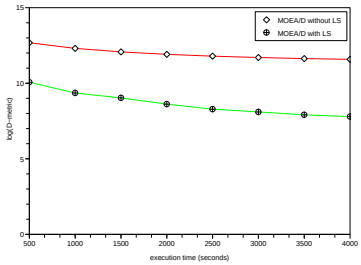

Figure 9: D-metric vs. execution time in MOEA/D with and without local search for the instance with 1500 cities. The scale of the D-metric is log.

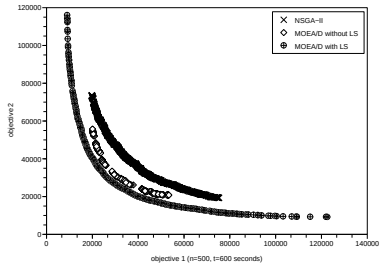

Figure 10: Distribution of the final solutions generated by the three algorithm generated for the instance with 500 cities.

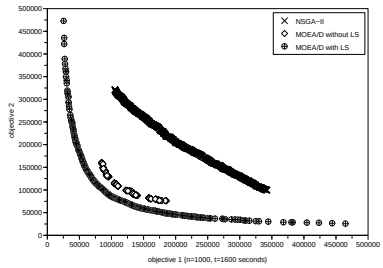

Figure 11: Distribution of the final solutions generated by the three algorithm generated for the instance with 1000 cities.

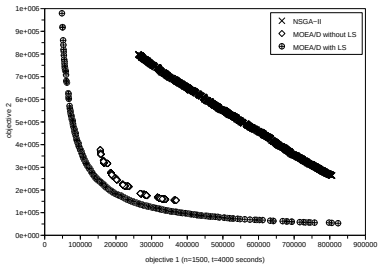

Figure 12: Distribution of the final solutions generated by the three algorithm generated for the instance with 1500 cities.