How to Choose Solutions for Local Search in Multiobjective Combinatorial Memetic Algorithms

Hisao Ishibuchi, Yasuhiro Hitotsuyanagi, Yoshihiko Wakamatsu and Yusuke Nojima

Department of Computer Science and Intelligent Systems, Graduate School of Engineering, Osaka Prefecture University, 1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531, Japan
{hisaoi@, hitotsu@ci., wakamatsu@ci., nojima@}cs.osakafu-u.ac.jp

Abstract. This paper demonstrates that the performance of multiobjective memetic algorithms (MOMAs) for combinatorial optimization strongly depends on the choice of solutions to which local search is applied. We first examine the effect of the tournament size to choose good solutions for local search on the performance of MOMAs. Next we examine the effectiveness of an idea of applying local search only to non-dominated solutions in the offspring population. We show that this idea has almost the same effect as the use of a large tournament size because both of them lead to high selection pressures. Then we examine different configurations of genetic operators and local search in MOMAs. For example, we examine the use of genetic operators after local search. In this case, improved solutions by local search are used as parents for recombination while local search is applied to the current population after generation update.

Keywords: Multiobjective genetic local search (MOGLS), evolutionary multiobjective optimization (EMO), hybrid algorithms, memetic algorithms, multiobjective combinatorial optimization.

1 Introduction

Since the mid-1990s [3], [4], local search has often been combined with evolutionary multiobjective optimization (EMO) algorithms to improve their search ability in the literature [11]. Hybrid EMO algorithms with local search were first proposed under the name of multiobjective genetic local search (MOGLS [3], [4], [7], [8]). Such a hybrid algorithm is also referred to as a multiobjective memetic algorithm (MOMA [6], [9]-[11]). In early studies [3], [4], [7]-[10], MOMAs were mainly applied to multiobjective combinatorial optimization problems. Recently local search has been also combined with EMO algorithms for multiobjective continuous optimization [13].

It is well-known that hybrid evolutionary algorithms with local search have high search ability for single-objective combinatorial optimization problems. They are often called genetic local search (GLS) or memetic algorithms (MAs [14]). A number of issues for designing high-performance MAs have been discussed for single-objective optimization [12], [16], [17] and multiobjective optimization [5]. An important issue is the balance between local search and genetic search especially in MO-

MAs [6]. When this balance is not appropriately specified, the performance of EMO algorithms is often severely degraded by the hybridization with local search.

Another important issue in the design of high-performance MOMAs is the choice of solutions to which local search is applied. This issue has not been discussed in detail in MOMAs for multiobjective combinatorial optimization in the literature. This is because the performance of EMO algorithms is usually improved by simply applying local search to good offspring as long as the balance between local search and genetic search is appropriate. In this paper, we examine a number of strategies for choosing local search solutions in MOMAs. For this purpose, we use a simple MOMA called S-MOGLS [2], which is a hybrid algorithm of NSGA-II [1] with local search. Its generation update mechanism is illustrated in Fig. 1. First genetic search (i.e., selection, crossover and mutation) is applied to the current population in the same manner as NSGA-II. Next local search is applied to the offspring population. Then the next population is constructed by choosing good solutions from the current, offspring and improved populations in the same manner as NSGA-II. Of course, similar MOMAs can be designed using other EMO algorithms instead of NSGA-II.

In this paper, we examine the following strategies to choose local search solutions:

(i) Selection of local search solutions from the offspring population as in Fig. 1.
(ii) Selection from non-dominated solutions in the offspring population.
(iii) Selection from a merged population of the current and offspring populations.
(vi) Selection from the current population. In this case, local search is used before genetic search as shown in Fig. 2.

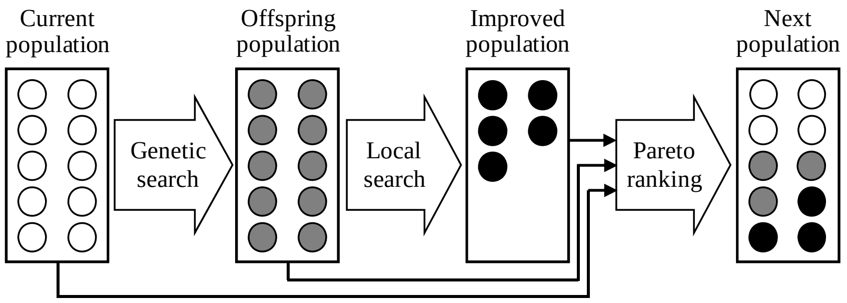


Fig. 1. Our standard MOMA with the CP-GS-LS structure (S-MOGLS [2]).

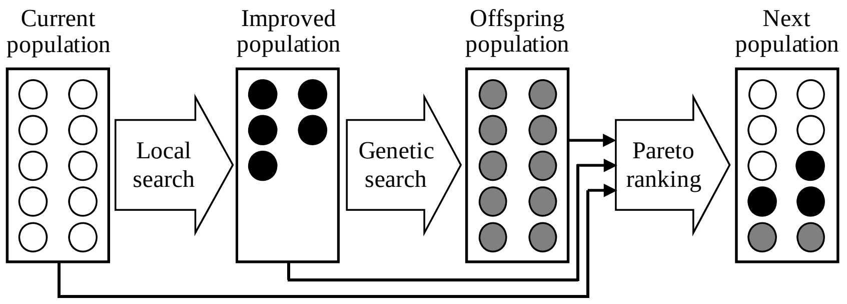


Fig. 2. Our MOMA with the CP-LS-GS structure.

2 Our Multiobjective Memetic Algorithm

Let us consider the following *k*-objective maximization problem:

$$\text{Maximize } \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}),\ f_2(\mathbf{x}),\ ...,\ f_k(\mathbf{x}))\ . \tag{1}$$

We explain our MOMA and its variants using this multiobjective problem.

Our MOMA in Fig. 1 is a simple hybrid algorithm of NSGA-II with local search. We denote the structure of MOMAs in Fig. 1 as “CP-GS-LS” since local search (LS) is used after genetic search (GS) is applied to the current population (CP). The outline of our MOMA with the CP-GS-LS structure can be written as follows:

[MOMA with the CP-GS-LS structure]

```
Step 1: $P$ = Initialize($P$)
Step 2: While the stopping condition is not satisfied, do
Step 3:   $P'$ = Genetic Search($P$)
Step 4:   $P''$ = Local Search($P'$)
Step 5:   $P$ = Generation Update($P \cup P' \cup P''$)
Step 6: End while
Step 7: Return Non-dominated($P$)
```

First an initial population P with N_{pop} solutions is randomly generated in Step 1 where N_{pop} is the population size. Then Steps 3-5 are iterated until a prespecified stopping condition is satisfied. Step 3 is exactly the same as the genetic search of NSGA-II. An offspring population P' is generated. Step 5 is conceptually the same as the generation update mechanism of NSGA-II. The best N_{pop} solutions are selected as the next population P from the merged population $P \cup P' \cup P''$ in Step 5 using Pareto ranking and crowding distance.

In Step 4, we use the following weighted sum fitness function for local search:

$$f(\mathbf{x}) = \lambda_1 f_1(\mathbf{x}) + \lambda_2 f_2(\mathbf{x}) + \cdots + \lambda_k f_k(\mathbf{x})\,, \tag{2}$$

where $\boldsymbol{\lambda}=(\lambda_1, \lambda_2, ..., \lambda_k)$ is a weight vector. Of course we can use other functions. We use a set of uniformly distributed weight vectors satisfying the following conditions:

$$\lambda_1 + \lambda_2 + \cdots + \lambda_k = d \quad \text{and} \quad \lambda_i \in \{0, 1, ..., d\} \text{ for } i = 1, 2, ..., k\ . \tag{3}$$

The same weight vector generation mechanism was used in [15] and [18]. We specify d in (3) as $d = 100$ to generate 101 weight vectors for two-objective problems.

In Step 4, first a weight vector is randomly drawn from the weight vector set. Then a local search solution is selected from the offspring population P' using tournament selection with replacement. Various values of tournament size are examined in our computational experiments. Each solution in P' is evaluated by the weighted sum fitness function in (2) with the current weight vector. Local search is applied to the chosen solution. The weighted sum fitness function in (2) with the current weight vector is used to compare the current solution and its neighbors in local search.

In local search, a neighbor is randomly generated from the current solution. When a better neighbor is found, the current solution is replaced with it. That is, we use the first move strategy where local search accepts the first improved neighbor rather than the best move strategy. As a termination condition of local search, we use the total number of examined neighbors (say, N_{LS} neighbors) in a series of local search from the local search solution (i.e., starting solution) chosen from the offspring population.

The number of solutions to which local search is applied in each generation can be specified using the local search application probability P_{LS} as $P_{\mathrm{LS}} N_{\mathrm{pop}}$. Since N_{LS} neighbors are examined in a series of local search from each local search solution, the total number of examined solutions by local search in each generation can be calculated as $P_{\mathrm{LS}} N_{\mathrm{pop}} N_{\mathrm{LS}}$ while N_{pop} solutions are examined by genetic search.

3 Variants of Our Multiobjective Memetic Algorithm

As shown in Fig. 1, local search is usually applied to the offspring population in MOMAs. This is, however, not necessarily the best structure of MOMAs. In general, the offspring population may include many poor solutions due to the random nature of crossover and mutation. In Fig. 3, we show an example of the current population and its offspring population in a single run of NSGA-II on the two-objective 500-item 0/1 knapsack problem [19]. Conditions of this experiment will be shown in Section 4.

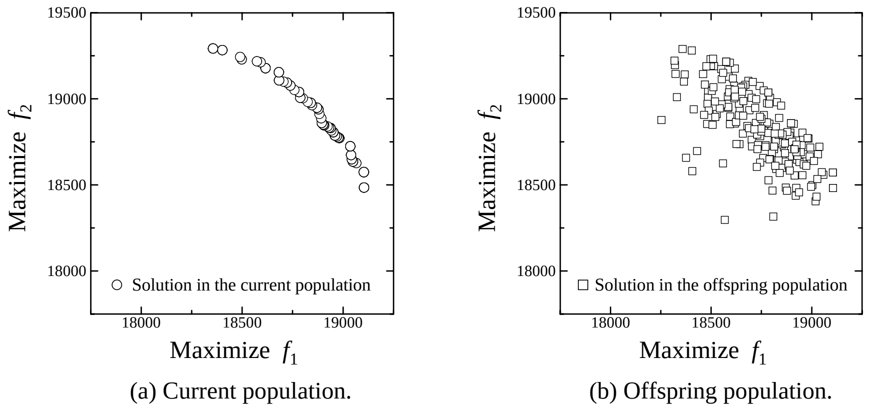


Fig. 3. Current and offspring populations at the 200th generation in a single run of NSGA-II on the two-objective 500-item 0/1 knapsack problem (see Section 4 for parameter values).

The application of local search to poor solutions is often the waste of time. In our MOMA, we can choose a good solution from the offspring population using tournament selection with a large tournament size. We can also use a strategy to apply local search only to non-dominated solutions in the offspring population. This idea is written as follows in our MOMA in the previous session.

```
Step 4:    $P''$ = Local Search(Non-dominated($P'$))
```

As shown in Fig. 3 (a), the current population does not include many poor solutions. This is because the deterministic generation update mechanism of NSGA-II always chooses the best N_{pop} solutions for the next population. It may be a good idea to apply local search to the current population. One possible implementation of this idea is to choose local search solutions from the current and offspring populations. We denote this version of our MOMA as "(CP-GS)-LS" in order to explicitly show that LS is applied to the current and offspring populations. Except for the selection of local search solutions, the (CP-GS)-LS structure is the same as the CP-GS-LS structure in Fig. 1. Thus only Step 4 of our MOMA algorithm with the CP-GS-LS structure in Fig. 1 is modified for describing the (CP-GS)-LS structure as follows:

[MOMA with the (CP-GS)-LS structure]

```
Step 4:   $P'' = $ Local Search($P \cup P'$)
```

It is also possible to use local search before genetic search in each generation as shown in Fig. 2 in order to apply local search to solutions in the current population. We denote this version as "CP-LS-GS". In the CP-LS-GS structure, parents for recombination are chosen from the improved population. The difference between the CP-LS-GS structure in Fig. 2 and the CP-GS-LS structure in Fig. 1 is only the order of local search (LS) and genetic search (GS). Thus only Step 3 and Step 4 of our MOMA algorithm with the CP-GS-LS structure in Fig. 1 are modified as follows:

[MOMA with the CP-LS-GS structure]

```
Step 3:   $P' = $ Local Search($P$)
Step 4:   $P'' = $ Genetic Search($P'$)
```

One potential difficulty in the CP-LS-GS structure in Fig. 2 is the possibility that the improved population *P*' is empty (i.e., no solutions are improved by local search in Step 3). Only in this special case, we choose parents for recombination in genetic search from the current population *P*. This potential difficulty of the CP-LS-GS structure can be easily removed by choosing parents for recombination from the current and improved populations, which leads to the (CP-LS)-GS structure as follows:

[MOMA with the (CP-LS)-GS structure]

```
Step 3:   $P' = $ Local Search($P$)
Step 4:   $P'' = $ Genetic Search($P \cup P'$)
```

4 Computational Experiments

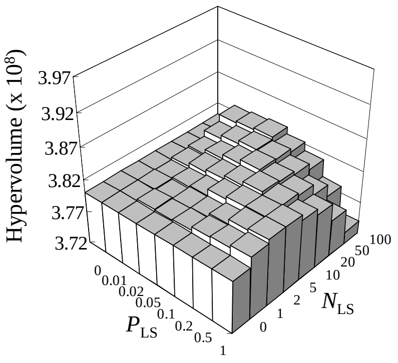
In this section, we examine the effect of the choice of local search solutions on the performance of our MOMA through computational experiments.

Knapsack Problem: We first show experimental results on the two-objective 500-item knapsack problem [19]. Our experiments were performed under the following setting (we used the same greedy repair as in [19] to handle infeasible solutions):

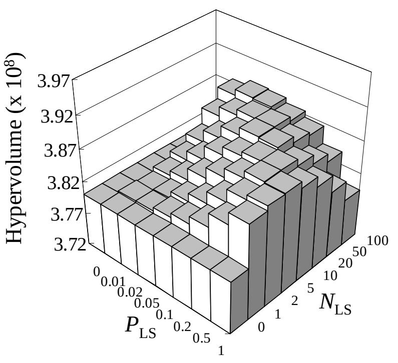
Population size: 200,
Total number of examined solutions (Termination conditions): 400,000,
Tournament size for parent selection in genetic search: 2,

Crossover probability in genetic search: 0.8 (Uniform crossover),
Mutation probability in genetic search: 0.002 (Bit-flip mutation),
Tournament size for local search solution selection: 1, 2, 5, 10, 20, 50,
Neighbor generation in LS: Bit-flip operation with the probability 0.008,
Local search probability: $P_{LS} = 0, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0$,
LS length (LS termination condition): $N_{LS} = 0, 1, 2, 5, 10, 20, 50, 100$.

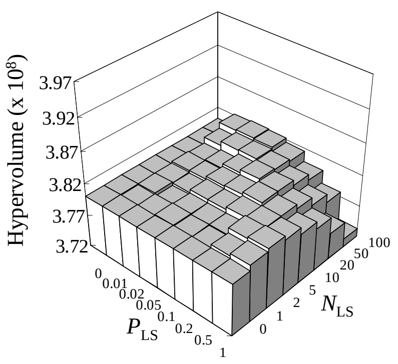
In Fig. 4, we show average hypervolume values over 100 runs by our standard CP-GS-LS MOMA and its non-dominated variant (i.e., selection of only non-dominated offspring for local search). We used the origin (0, 0) of the objective space as the reference point for hypervolume calculation. Since our MOMA is exactly the same as NSGA-II when local search is not used (i.e., when $P_{LS} = 0$ or $N_{LS} = 0$), the left-bottom and left-top rows with the same height bars (about 3.8×10^8 hypervolume) can be viewed as the results of NSGA-II in each plot. We obtained better results from the two variants of our CP-GS-LS MOMA than NSGA-II in a wide range of P_{LS} and N_{LS}. Their performance was, however, severely degraded when both P_{LS} and N_{LS} were too large (i.e., when the genetic search and local search balance was not good).



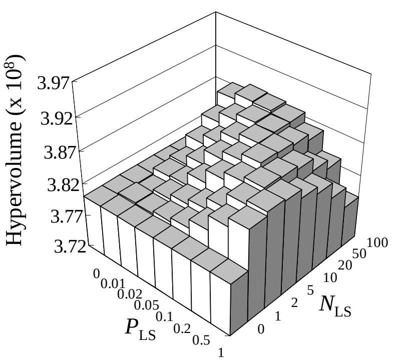
(a) Standard CP-GS-LS MOMA (Size 10).



(b) Standard CP-GS-LS MOMA (Size 50).



(c) Non-dominated CP-GS-LS (Size 2).

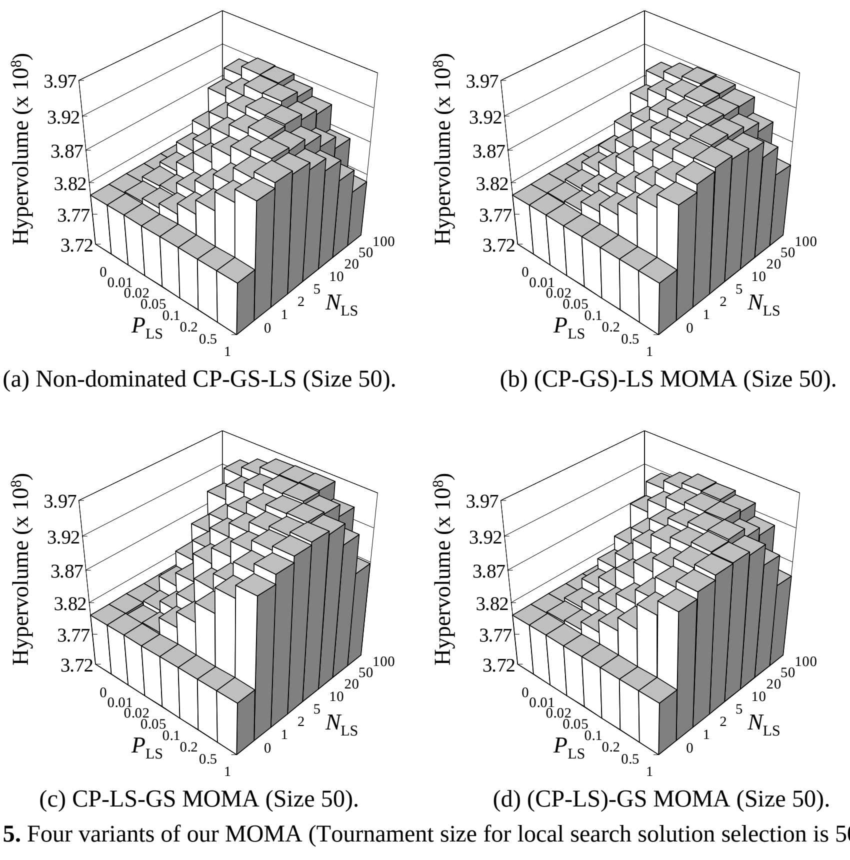


(d) Non-dominated CP-GS-LS (Size 10).

Fig. 4. Two variants of CP-GS-LS (Size: tournament size for local search solution selection).

In Fig. 4, similar results were obtained by the two variants of the CP-GS-LS structure. It should be noted, however, that larger values were used as tournament size in the upper plots than the lower plots. This means that the use of non-dominated solutions for local search in the lower plots has a similar effect to the use of large tournament size for local search solution selection on the performance of our MOMA.

Four variants of our MOMA are compared with each other in Fig. 5 under the same tournament size of 50 for local search solution selection in all the four plots. The best results were obtained from the CP-LS-GS MOMA in Fig. 5 (c).



(a) Non-dominated CP-GS-LS (Size 50). (b) (CP-GS)-LS MOMA (Size 50).

(c) CP-LS-GS MOMA (Size 50). (d) (CP-LS)-GS MOMA (Size 50).

Fig. 5. Four variants of our MOMA (Tournament size for local search solution selection is 50).

In order to visually demonstrate the statistical significance of the difference in the performance between the CP-GS-LS and CP-LS-GS structures, we show the histogram of 100 hypervolume values obtained from 100 runs of each variant in Fig. 6. In Fig. 6, we used the experimental results with the best combination of P_{LS} and N_{LS} with respect to the average hypervolume in each plot of Fig. 5. For comparison, we also show the results of NSGA-II. We can observe in Fig. 6 that the CP-LS-GS variant clearly outperformed the standard CP-GS-LS MOMA. We can also see that the hybridization with local search clearly improved the performance of NSGA-II.

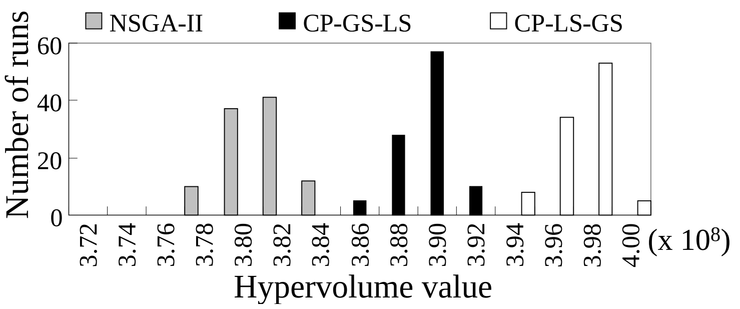


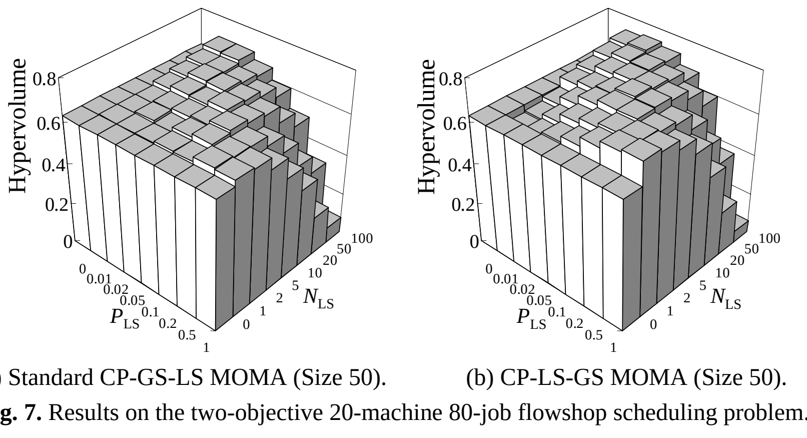
Fig. 6. Histogram of 100 hypervolume values obtained from 100 runs of each algorithm.

Flowshop Scheduling: We also applied our MOMA variants to two-objective 20-machine flowshop scheduling problems with 20 and 80 jobs [6]. We used the following setting (The other parameters were the same as in the previous experiments):

Total number of examined solutions (Termination conditions): 100,000,
Crossover probability in genetic search: 0.9 (Two-point crossover [6]),
Mutation probability in genetic search: 0.6 (Insertion mutation [6]),
Neighbor generation in local search: A single use of an insertion operator.

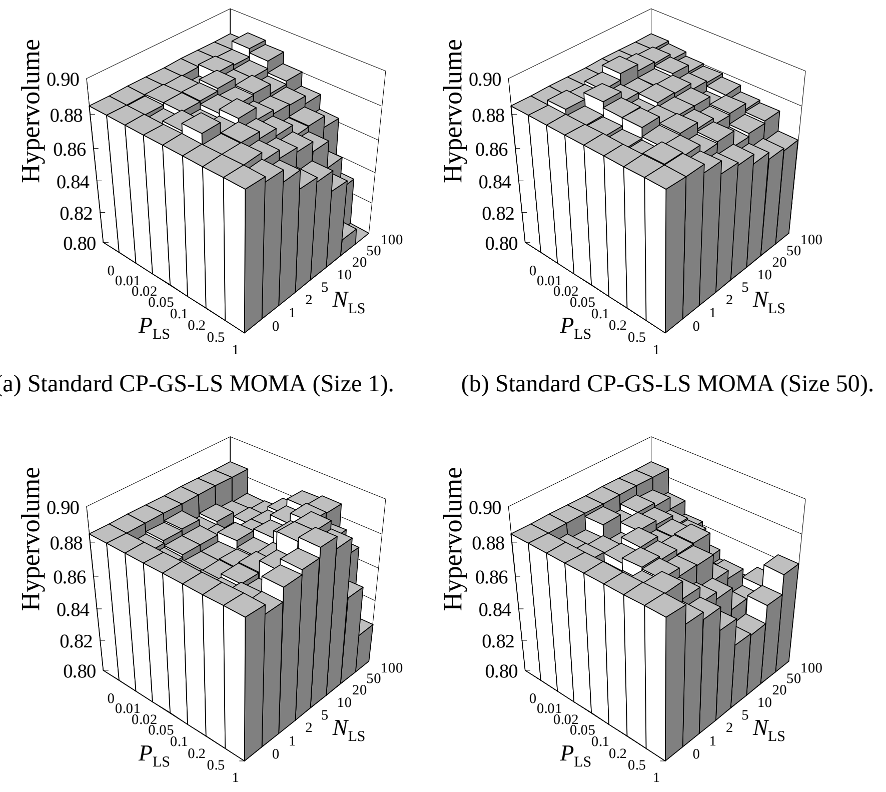
Before hypervolume calculation, we normalized the objective space so that overall non-dominated solutions were in the unit square $[0, 1] \times [0, 1]$. Hypervolume was calculated in the normalized objective space using the reference point (1.1, 1.1). Due to the page limitation, we show a part of experimental results on the 80-job problem in Fig. 7. The best results were obtained from the CP-LS-GS structure with the largest tournament size in Fig. 7 for the 80-job problem as in Fig. 5 on the knapsack problem.

In Fig. 8, we show experimental results on the 20-job problem. The size of the search space of the 20-job problem is 20!, which is much smaller than 80! of the 80-job problem. Thus good results were not obtained from high selection pressure for local search solution selection. The best results were obtained from CP-LS-GS with the tournament size 1 (i.e., random selection) for local search solution selections.



(a) Standard CP-GS-LS MOMA (Size 50). (b) CP-LS-GS MOMA (Size 50).

Fig. 7. Results on the two-objective 20-machine 80-job flowshop scheduling problem.



(a) Standard CP-GS-LS MOMA (Size 1). (b) Standard CP-GS-LS MOMA (Size 50).

(c) CP-LS-GS MOMA (Size 1). (d) CP-LS-GS MOMA (Size 50).

Fig. 8. Results on the two-objective 20-machine 20-job flowshop scheduling problem.

5 Conclusions

We demonstrated that the choice of local search solutions had a large effect on the performance of our MOMA, which is a hybrid algorithm of NSGA-II with local search. Its performance was improved by choosing good solutions for local search through tournament selection with large tournament size. Among four variants of our MOMA, the best results were obtained from a non-standard structure of MOMA: CP-LS-GS. These observations were obtained from our computational experiments on a two-objective 500-item knapsack problem and a two-objective 80-job flowshop scheduling problem. Whereas we did not report due to the page limitation, similar results were also obtained from computational experiments on a three-objective 80-job flowshop problem and 500-item knapsack problems with four and six objectives. The best results, however, were obtained from random selection of local search solutions for a small-size flowshop problem with 20 jobs. Even in this case, the CP-LS-GS structure was the best among the four variants. These observations suggest high potential of the CP-LS-GS structure which has not been examined in many studies in the literature.

References

1. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Trans. on Evolutionary Computation 6 (2002) 182-197
2. Ishibuchi, H., Hitotsuyanagi, Y., Tsukamoto, N., Nojima, Y.: Use of Heuristic Local Search for Single-Objective Optimization in Multiobjective Memetic Algorithms. Lecture Notes in Computer Science, Vol. 5199: PPSN X. Springer, Berlin (2008) 743-752
3. Ishibuchi, H., Murata, T.: Multi-Objective Genetic Local Search Algorithm. Proc. of 1996 IEEE International Conference on Evolutionary Computation (1996) 119-124
4. Ishibuchi, H., Murata, T.: A Multi-Objective Genetic Local Search Algorithm and Its Application to Flowshop Scheduling. IEEE Trans. on Systems, Man, and Cybernetics - Part C: Applications and Reviews 28 (1998) 392-403
5. Ishibuchi, H., Narukawa, K.: Some Issues on the Implementation of Local Search in Evolutionary Multiobjective Optimization. Lecture Notes in Computer Science, Vol. 3102: GECCO 2004. Springer, Berlin (2004) 1246-1258
6. Ishibuchi, H., Yoshida, T., Murata, T.: Balance between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling. IEEE Trans. on Evolutionary Computation 7 (2003) 204-223
7. Jaszkiewicz, A.: Genetic Local Search for Multi-Objective Combinatorial Optimization. European Journal of Operational Research 137 (2002) 50-71
8. Jaszkiewicz, A.: On the Performance of Multiple-Objective Genetic Local Search on the 0/1 Knapsack Problem - A Comparative Experiment. IEEE Trans. on Evolutionary Computation 6 (2002) 402-412
9. Knowles, J. D., Corne, D. W.: M-PAES: A Memetic Algorithm for Multiobjective Optimization. Proc. of 2000 IEEE Congress on Evolutionary Computation (2000) 325-332
10. Knowles, J. D., Corne, D. W.: A Comparison of Diverse Approaches to Memetic Multiobjective Combinatorial Optimization. Proc. of 2000 Genetic and Evolutionary Computation Conference Workshop Program: WOMA I (2000) 103-108
11. Knowles, J. D., Corne, D. W.: Memetic Algorithms for Multiobjective Optimization: Issues, Methods and Prospective. In Hart, W. E., Krasnogor, N., Smith, J. E. (eds.): Recent Advances in Memetic Algorithms, Springer, Berlin (2005) 313-352
12. Krasnogor, N., Smith, J.: A Tutorial for Competent Memetic Algorithms: Model, Taxonomy, and Design Issues. IEEE Trans. on Evolutionary Computation 9 (2005) 474-488
13. Lara, A., Sanchez, G., Coello, C. A. C., Schutze, O.: HCS: A New Local Search Strategy for Memetic Multiobjective Evolutionary Algorithms. IEEE Trans. on Evolutionary Computation 14 (2010) 112-132
14. Moscato, P.: Memetic Algorithms: A Short Introduction. In Corne, D., Dorigo, M., Glover, F. (eds.): New Ideas in Optimization. McGraw-Hill, London (1999) 219-234
15. Murata, T., Ishibuchi, H., Gen, M.: Specification of Genetic Search Directions in Cellular Multi-Objective Genetic Algorithm. Lecture Notes in Computer Science, Vol. 1993: EMO 2001, Springer, Berlin (2001) 82-95
16. Ong, Y. S., Keane, A. J.: Meta-Lamarckian Learning in Memetic Algorithms. IEEE Trans. on Evolutionary Computation 8 (2004) 99-110
17. Ong, Y. S., Lim, M. H., Zhu, N., Wong, K. W.: Classification of Adaptive Memetic Algorithms: A Comparative Study. IEEE Trans. on Systems, Man, and Cybernetics: Part B - Cybernetics 36 (2006) 141-152
18. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Trans. on Evolutionary Computation 11 (2007) 712-731
19. Zitzler, E., Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. IEEE Trans. on Evolutionary Computation 3 (1999) 257-271