Hybrid Evolutionary Multi-Objective Optimization Algorithms

Hisao ISHIBUCHI and Tadashi YOSHIDA
Department of Industrial Engineering, Osaka Prefecture University
1-1 Gakuen-cho, Sakai, Osaka 599-8531, Japan

Abstract. This paper examines how the search ability of evolutionary multi-objective optimization (EMO) algorithms can be improved by the hybridization with local search through computational experiments on multi-objective permutation flowshop scheduling problems. The task of EMO algorithms is to find a variety of non-dominated solutions of multi-objective optimization problems. First we describe our multi-objective genetic local search (MOGLS) algorithm, which is the hybridization of a simple EMO algorithm with local search. Next we discuss some implementation issues of local search in our MOGLS algorithm such as the choice of initial (i.e., starting) solutions for local search and a termination condition of local search. Then we implement hybrid EMO algorithms using well-known EMO algorithms: SPEA and NSGA-II. Finally we compare those EMO algorithms with their hybrid versions through computational experiments. Experimental results show that the hybridization with local search can improve the search ability of the EMO algorithms when local search is appropriately implemented in their hybrid versions.

1. Introduction

Since Schaffer's work [1], evolutionary algorithms have been applied to various multi-objective optimization problems for finding their Pareto-optimal solutions. Evolutionary algorithms for multi-objective optimization are often referred to as EMO (evolutionary multi-objective optimization) algorithms. For review of this field, see [2], [3]. The task of EMO algorithms is to find Pareto-optimal solutions as many as possible. It is, however, impractical to try to find true Pareto-optimal solutions of large-scale combinatorial optimization problems. Thus non-dominated solutions among examined ones are presented to decision makers as a result of the search by EMO algorithms. In this case, EMO algorithms try to drive populations to true Pareto-optimal solutions as close as possible for obtaining a variety of near Pareto-optimal solutions.

One promising approach for improving the search ability of EMO algorithms to find near Pareto-optimal solutions is the hybridization with local search. The hybridization of evolutionary algorithms with local search has already been investigated in many studies for single-objective optimization problems [4], [5]. Such a hybrid algorithm is often referred to as memetic algorithms. A hybrid evolutionary algorithm with local search for multi-objective optimization was first implemented in [6], [7] as a multi-objective genetic local

search (MOGLS) algorithm. Jaszkiewicz [8] improved the performance of the MOGLS algorithm by modifying the selection mechanism for choosing parent solutions for crossover in its EMO part. The performance of the MOGLS algorithm was also improved by introducing a selection mechanism into its local search part for choosing good solutions to which local search is applied in each generation [9], [10]. Knowles & Corne [11] combined their Pareto archived evolution strategy (PAES [12]) with a crossover operation for designing a memetic PAES (M-PAES). In their M-PAES, the Pareto-dominance relation and the grid-type partition of the multi-dimensional objective space were used for determining the acceptance (or rejection) of new solutions generated in genetic search and local search. The M-PAES had a special form of elitism inherent in the PAES.

Generic frameworks of hybrid EMO algorithms are shown in Fig. 1. The two frameworks in Fig. 1 are the same except for the order of genetic search and local search. Genetic operations are first applied to an initial population in Fig. 1 (a). On the other hand, genetic operations are applied after an initial population is improved by local search in Fig. 1 (b). The two frameworks are executed in the same manner after the second generation: the EMO part and the local search part are iterated for finding Pareto-optimal solutions. Emphasis is implicitly placed on the local search part in Fig. 1 (b) while the EMO part is implicitly viewed as the main part in Fig. 1 (a). In this paper, we use the framework in Fig. 1 (a) for describing hybrid EMO algorithms. In Fig. 1 (a), the local search part can be also viewed as a special kind of mutation in EMO algorithms.

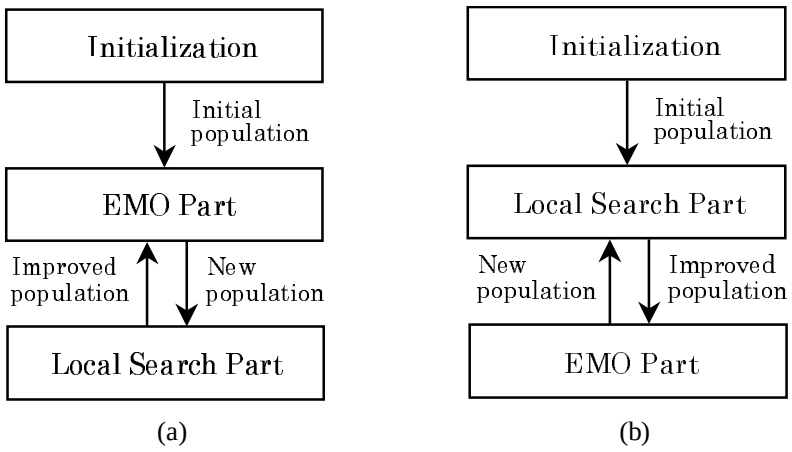

Fig. 1. Two generic frameworks of hybrid EMO algorithms. In (a), local search is applied to new solutions generated by the EMO part. In (b), genetic operations are applied to an improved population generated by the local search part.

In this paper, we first rewrite the MOGLS algorithm in our former studies [9], [10] by introducing a new parameter: local search probability P_{LS}. Local search is probabilistically applied to each solution with the local search probability P_{LS} in our MOGLS algorithm. Next we discuss some implementation issues of local search in our MOGLS algorithm such as the choice of initial (i.e., starting) solutions for local search and a termination condition of local search. In the local search part of our MOGLS algorithm, good solutions are selected from the current population as initial solutions for local search. Then local search is probabilistically applied to each of the selected solutions. For decreasing the CPU time spent by local search (i.e., for striking a balance between genetic search and local search

[10]), we use an early termination strategy where local search is terminated before finding a locally optimal solution. The probabilistic application of local search and its early termination are for preventing a possible negative effect of local search: It may cause a premature convergence to local solutions. Then we implement hybrid versions of well-known EMO algorithms: SPEA (strength Pareto evolutionary algorithm [13]) and NSGA-II (revised non-dominated sorting genetic algorithm [14]). The SPEA and the NSGA-II are combined with local search in the framework shown in Fig. 1 (a). Finally we compare those EMO algorithms with their hybrid versions through computational experiments on multi-objective permutation flowshop scheduling problems.

2. Multi-Objective Genetic Local Search Algorithm

Let us consider the following *n*-objective minimization problem:

$$\text{Minimize } \mathbf{z} = (f_1(\mathbf{x}),\ f_2(\mathbf{x}), ...,\ f_n(\mathbf{x})), \tag{1}$$

$$\text{subject to } \mathbf{x} \in \mathbf{X}, \tag{2}$$

where $\mathbf{z}$ is the objective vector, $f_i(\mathbf{x})$ is the *i*-th objective to be minimized, $\mathbf{x}$ is the decision vector, and $\mathbf{X}$ is the feasible region in the decision space.

The EMO part of our MOGLS algorithm is a simple multi-objective genetic algorithm with the roulette wheel selection based on a scalar fitness function. It has a secondary population for elitism [15]. As in standard single-objective genetic algorithms, first an initial population of N_{pop} solutions is randomly generated where N_{pop} is the population size. Non-dominated solutions in the initial population are identified, and a secondary population is constructed from their copies. The fitness value of each solution in the current population is evaluated using the following scalar fitness function (to be minimized):

$$f(\mathbf{x}) = w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \cdots + w_n f_n(\mathbf{x}), \tag{3}$$

where w_i is a normalized non-negative random weight for the *i*-th objective function $f_i(\mathbf{x})$: $w_1 + w_2 + \cdots + w_n = 1$ and $w_i \geq 0$ for $\forall i$. A pair of parent solutions is selected from the current population using the roulette wheel selection scheme with the linear scaling:

$$P_{\text{S}}(\mathbf{x}) = \frac{f_{\max}(\Psi) - f(\mathbf{x})}{\sum_{\mathbf{y} \in \Psi} \{ f_{\max}(\Psi) - f(\mathbf{y}) \}}, \tag{4}$$

where $P_{\text{S}}(\mathbf{x})$ is the selection probability of the solution $\mathbf{x}$, Ψ denotes the current population, and $f_{\max}(\Psi)$ is the largest (i.e., worst) fitness value $f(\mathbf{x})$ in the current population Ψ under the current weight vector $\mathbf{w} = (w_1, w_2, ..., w_n)$. For finding a variety of non-dominated solutions (i.e., for realizing various search directions in the *n*-dimensional objective space), the weight vector is randomly specified whenever a pair of parent solutions is to be selected. That is, the selection of each pair of parent solutions is governed by a different weight vector. A pre-specified number of pairs of parent solutions are selected in this manner. New solutions are generated by crossover and mutation in the same manner as standard single-objective genetic algorithms. Then a pre-specified number of non-dominated solutions are randomly selected from the secondary population as elite solutions, and their copies are added to the newly generated solutions. In this manner, a new population is constructed from the solutions generated by the genetic operations and the copies of non-dominated solutions randomly selected from the secondary population.

Next the local search part receives the new population generated in the EMO part. In our MOGLS algorithm, an initial (i.e., starting) solution for local search is selected from the current population (i.e., population generated in the EMO part) using the scalar fitness function in (3) with random weight values. Weight values are randomly specified whenever an initial solution is to be selected for local search. That is, each initial solution is selected based on a different weight vector. Local search for the selected initial solution is governed by the scalar fitness function with the weight vector used in its selection. The tournament selection with the tournament size five is used in our computational experiments for the selection of an initial solution for local search. Local search is probabilistically applied to a copy of the selected initial solution with the local search probability P_{LS}. When local search is not applied, a copy of the selected initial solution is added to the next population. When local search is applied, a neighbor of the current solution is randomly selected. If the neighbor is superior to the current solution with respect to the scalar fitness function with the current weight vector, the current solution is immediately replaced with the neighbor. That is, we use the hill-climbing algorithm with the first improvement strategy in the local search part instead of the best improvement (i.e., steepest hill-climbing) strategy. For preventing the local search part from spending almost all the available CPU time, we restrict the number of successive fails of local move to k where k is a user-definable parameter. This means that local search for the current solution is terminated when a better solution is not found among k neighbors randomly selected from the neighborhood of the current solution. Of course, when a better neighbor is found, we can examine k neighbors of the new current solution again. When local search is terminated, the current solution is added to the new population. The selection of an initial solution from the current population and the probabilistic application of local search are iterated N_{pop} times for generating the new population with N_{pop} solutions (i.e., for improving the population generated by the EMO part).

The outline of our MOGLS algorithm can be written as follows:

MOGLS Algorithm

Step 0) Initialization: Randomly generate an initial population of N_{pop} solutions.

[EMO Part]

Step 1) Evaluation: Calculate the n objectives for each solution in the current population. Then update the secondary population where non-dominated solutions are stored separately from the current population.

Step 2) Selection: Repeat the following procedures to select $(N_{\mathrm{pop}} - N_{\mathrm{elite}})$ pairs of parent solutions where N_{elite} is the number of elite solutions.

(a) Randomly specify the weight values $w_1, w_2, \ldots, w_n$.

(b) Select a pair of parent solutions using the scalar fitness function in (3). The roulette wheel scheme in (4) is used for the selection of parent solutions.

Step 3) Crossover and mutation: Apply a crossover operation to each of the selected $(N_{\mathrm{pop}} - N_{\mathrm{elite}})$ pairs of parent solutions. A new solution is generated from each pair. Then apply a mutation operation to each of the generated new solutions.

Step 4) Elitist strategy: Randomly select N_{elite} solutions from the secondary population. Then add their copies to the $(N_{\mathrm{pop}} - N_{\mathrm{elite}})$ solutions generated in Step 3 to construct a population of N_{pop} solutions.

[Local Search Part]

Step 5) Local search: Iterate the following three steps N_{pop} times. Then replace the

current population with N_{pop} solutions obtained by the following steps.

(a) Randomly specify the weight values $w_1, w_2, ..., w_n$.

(b) Select a solution from the current population using tournament selection with replacement based on the scalar fitness function in (3) with the current weight vector specified in (a). A copy of the selected solution is used in (c). Thus no solution is removed from the current population.

(c) Apply local search to a copy of the selected solution using the current weight vector with the local search probability P_{LS}. When local search is applied, the current solution after local search is included in the next population. As mentioned above, local search is terminated when no better solution is found among *k* neighbors randomly generated from the current solution. On the other hand, when local search is not applied, a copy of the selected solution in (b) is added to the next population.

Step 6) Return to Step 1.

This algorithm is terminated when a pre-specified stopping condition is satisfied. In this paper, we use the number of examined solutions as the stopping condition for comparing different algorithms under the same computation load.

Let us demonstrate how the performance of the EMO algorithm can be improved by the hybridization with local search in our MOGLS algorithm. As test problems, we generated eight flowshop scheduling problems in the same manner as [7]. The processing time of each job on each machine was specified as a random integer in the interval [1, 99]. The due date of each job was specified by adding a random integer in the interval [-100, 100] to its actual completion time in a randomly generated schedule. All the eight test problems have 20 machines. Using the number of objectives (*n*) and the number of jobs (*N*), we denote each of the eight test problems as *n*/*N* where $n = 2$, 3 and $N = 20$, 40, 60, 80. Four test problems have two objectives (i.e., $n = 2$): to minimize the makespan and to minimize the maximum tardiness. The other four test problems are three-objective problems (i.e., $n = 3$) with an additional objective: to minimize the total flow time.

In our computational experiments of this paper, we used the same genetic operations as [7]: the two-point order crossover and the shift change mutation. The shift change mutation is the same as the insertion operation: remove a randomly selected job and insert it into another position. The shift change mutation was also used in the local search part for generating a neighbor from the current solution as in [7]. We used the following parameter specifications in the EMO part:

Population size (N_{pop}): 60,
Crossover probability: 0.9,
Mutation probability per string: 0.6,
Number of elite solutions (N_{elite}): 10,
Stopping condition: Evaluation of 100,000 solutions.

Several different specifications were examined for the two parameters P_{LS} and *k*.

A solution set obtained by our MOGLS algorithm was evaluated by the average normalized distance from each reference solution (i.e., approximate Pareto-optimal solution) to the nearest solution in the obtained solution set. This performance criterion can measure the proximity of the obtained solution set to the Pareto front and the quality of the distribution of solutions in the obtained solution set. The reference solutions were found from ten independent runs of the MOGLS algorithm, the SPEA, and the NSGA-II for each

test problem with much longer CPU time (i.e., five million solutions were examined in each run of each algorithm). We compared 30 solution sets, which were obtained from ten runs of the three algorithms, with each other for finding non-dominated solutions. All the non-dominated solutions were used as reference solutions for each test problem. The objective space of each test problem was normalized so that the minimum and maximum values of each objective among the reference solutions became 0 and 100, respectively.

Using the reference solutions of each test problem with the normalized objective space, we evaluated solution sets obtained by the MOGLS algorithm with various specifications of the two parameters in the local search part: P_{LS} and k. We also evaluated the performance of the EMO part of the MOGLS algorithm. Average results over 50 independent runs are summarized in Table 1. From this table, we can see that the performance of the simple EMO algorithm was significantly improved by the hybridization with local search. We can also see that the performance of our MOGLS algorithm strongly depended on the two parameters: k and P_{LS}. The best result for each test problem in each table is indicated by "*". The best result for each test problem among all tables in this paper is underlined.

The performance of the MOGLS algorithm is significantly deteriorated when we remove the selection mechanism (i.e., selection of initial solutions for local search) from the local search part. We performed the same computational experiments as Table 1 using the MOGLS algorithm without the selection mechanism in the local search part. In this case, local search was probabilistically applied to every solution in the current population with the local search probability P_{LS} independent of their fitness values. When local search was applied to a solution $\mathbf{x}$, the local search direction for $\mathbf{x}$ (i.e., weight vector in the scalar fitness function) was specified using the concept of pseudo-weight vector [2]. The pseudo-weight w_i for the i-th objective $f_i(\mathbf{x})$ was defined for the solution $\mathbf{x}$ as

$$w_i = \frac{f_i^{\max} - f_i(\mathbf{x})}{f_i^{\max} - f_i^{\min}} \Bigg/ \sum_{j=1}^{n} \frac{f_j^{\max} - f_j(\mathbf{x})}{f_j^{\max} - f_j^{\min}}, \quad i = 1,2,...,n, \tag{5}$$

where $f_i^{\max}$ and $f_i^{\min}$ are the maximum and minimum values of the i-th objective $f_i(\mathbf{x})$ in the current population, respectively. It should be noted that the local search direction for each initial solution was specified by the weight vector used in the selection of that initial solution in the MOGLS algorithm with the selection mechanism in Table 1. Table 2 shows average results over 50 independent runs of the MOGLS algorithm without the selection mechanism. From the comparison between Table 1 and Table 2, we can see that the performance of the MOGLS algorithm was significantly deteriorated in Table 2 by removing the selection mechanism. That is, we can see that the selection of initial solutions for local search plays a significant role in the MOGLS algorithm. When the local search probability is very small (e.g., $P_{\mathrm{LS}} = 0.01$), the selection of initial solutions for local search can be viewed as the selection of good solutions from the current population for constructing the next population. The effect of such selection may be twofold: the improvement in the convergence speed to Pareto-optimal solutions and the decrease in the variety of solutions. Since our MOGLS algorithm uses a very simple EMO algorithm with the parent selection scheme based on the roulette wheel, the positive effect (i.e., the improvement in the convergence speed) dominates the negative effect (i.e., the decrease in the variety of solutions) in our computational experiments. As a result, the selection of initial solutions for local search significantly improved the performance of the MOGLS algorithm from Table 2 to Table 1 even when the local search probability was very small.

Table 1. Performance of the MOGLS algorithm with the selection mechanism in the local search part.

Test Problem	$P_{LS} = 0.01$			$P_{LS} = 0.05$			$P_{LS} = 0.1$			$P_{LS} = 1$			EMO Part
	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	
2/20	5.85	5.59	4.92*	6.14	5.67	4.94	6.23	5.97	4.93	5.89	5.75	5.39	42.61
2/40	22.11	20.65	18.68	21.17	19.27	19.01	20.15	18.50	19.43	18.01*	20.27	26.26	109.76
2/60	24.46	24.08	20.56	24.10	22.16	20.02*	23.64	21.88	20.89	22.55	23.41	27.69	107.52
2/80	110.28	89.50	70.67*	96.73	84.41	71.72	97.84	74.52	74.42	83.18	89.79	127.13	610.29
3/20	8.99	9.13	7.96*	8.96	8.66	8.25	9.05	8.44	8.42	9.45	9.14	9.50	50.06
3/40	22.07	21.15	20.19*	21.67	21.70	21.30	21.70	21.56	20.86	21.25	22.92	26.51	109.91
3/60	32.19	30.72	26.84	31.96	29.29	26.82*	30.35	29.00	27.18	30.52	32.23	35.91	130.88
3/80	36.93	35.00	33.18	35.98	34.93	33.93	34.17	34.91	32.84*	35.83	38.90	47.49	173.60

Table 2. Performance of the MOGLS algorithm without the selection mechanism in the local search part.

Test Problem	$P_{LS} = 0.01$			$P_{LS} = 0.05$			$P_{LS} = 0.1$			$P_{LS} = 1$			EMO Part
	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	
2/20	38.94	22.80	9.89	29.37	19.91	9.21	26.39	19.10	9.29	22.65	12.80	7.33*	42.61
2/40	100.28	66.53	39.13	80.56	56.47	37.64	67.54	53.76	37.70	47.41	43.64	34.69*	109.76
2/60	95.33	61.87	35.52	76.88	52.22	34.41	67.90	50.99	33.81*	46.30	41.09	33.97	107.52
2/80	565.8	390.5	181.3	479.4	338.0	170.3	428.9	305.9	161.6*	277.3	247.7	185.9	610.29
3/20	45.57	23.25	12.68	34.88	19.53	11.80	27.47	18.43	12.04	17.92	13.38	10.48*	50.06
3/40	101.15	60.13	32.68*	78.58	52.59	33.08	68.16	49.84	33.50	42.70	40.74	33.20	109.91
3/60	119.40	73.24	43.84	97.68	65.01	42.29	82.92	62.56	41.95	58.32	53.36	41.12*	130.88
3/80	161.62	97.92	54.98*	131.53	86.58	55.03	114.66	82.98	55.03	74.79	70.15	59.42	173.60

3. Hybrid SPEA and Hybrid NSGA-II

Since the local search part (i.e., Step 5 of our MOGLS algorithm) is independent of the EMO part, it can be combined with other EMO algorithms. In the hybridization with local search, we do not have to modify EMO algorithms. When EMO algorithms have a secondary population, it is updated after the current population is improved by local search. We implemented a hybrid SPEA and a hybrid NSGA-II by replacing the EMO part of the MOGLS algorithm with the SPEA and the NSGA-II, respectively. The hybridization was implemented in the framework of Fig. 1 (a). In those hybrid EMO algorithms, local search was applied to solutions in the primary population (i.e., it was not applied to the secondary population) as in the MOGLS algorithm.

In the same manner as Table 1, we applied the hybrid SPEA and the hybrid NSGA-II to the eight test problems. Average results over 50 independent runs are summarized in Table 3 and Table 4. In the computational experiments in Table 3 and Table 4, the selection mechanism was used in the local search part as in Table 1. In these tables, boldface shows that better results were obtained by the hybrid versions than the non-hybrid EMO algorithms. From Table 3, we can see that the performance of the SPEA was improved by the hybridization with local search for the two-objective test problems. On the other hand, we can see from Table 4 that the performance of the NSGA-II was improved by the hybridization for all test problems when the parameter specification was appropriate (e.g., $P_{LS} = 0.1$ and $k = 100$). It is very interesting to observe that the hybrid SPEA in Table 3 and the hybrid NSGA-II in Table 4 do not always outperform the MOGLS algorithm while their EMO parts clearly outperform the EMO part of our MOGLS algorithm in Table 1.

Table 3. Performance of the hybrid SPEA with the selection mechanism in the local search part.

Test Problem	$P_{LS} = 0.01$			$P_{LS} = 0.05$			$P_{LS} = 0.1$			$P_{LS} = 1$			SPEA
	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	
2/20	7.16	7.27	**5.22***	6.65	6.73	**5.39**	7.19	6.72	**5.40**	6.69	6.46	**5.49**	6.05
2/40	**17.24**	**17.60**	**14.94***	**17.14**	**17.01**	**16.67**	**17.11**	**16.93**	18.24	**16.83**	19.01	26.64	18.01
2/60	23.63	22.47	**21.17***	23.00	23.42	**21.59**	22.33	23.19	**21.27**	23.03	24.93	27.45	22.09
2/80	**80.02**	**76.38**	**52.84***	**80.38**	**71.04**	**54.78**	**76.22**	**66.85**	**62.99**	**76.92**	**72.87**	125.99	81.18
3/20	8.16	8.01	7.96	8.23	8.40	8.14	7.98	8.42	8.23	8.35	8.62	8.90	7.02*
3/40	19.17	18.26	19.10	18.90	19.42	19.53	18.74	19.70	19.79	19.41	21.60	25.03	15.81*
3/60	25.79	24.72	24.32	25.14	24.93	25.45	25.19	26.37	26.35	26.46	30.05	34.55	24.10*
3/80	32.21	31.91	30.85	31.91	33.12	31.63	31.81	32.99	32.25	33.64	37.02	46.57	28.10*

Table 4. Performance of the hybrid NSGA-II with the selection mechanism in the local search part.

Test Problem	$P_{LS} = 0.01$			$P_{LS} = 0.05$			$P_{LS} = 0.1$			$P_{LS} = 1$			NSGA-II
	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	
2/20	17.21	15.34	13.88	17.90	15.17	9.90	17.29	13.61	**8.33**	14.52	9.53	**5.60***	9.25
2/40	38.84	38.94	35.72	37.66	36.20	24.65	37.24	34.54	**19.83***	33.18	**21.23**	27.44	21.54
2/60	28.02	26.70	25.05	27.30	25.05	**21.60**	26.65	25.10	**21.24***	25.80	24.15	28.47	22.04
2/80	101.69	91.51	81.67	90.49	81.24	**70.19**	101.34	82.83	**68.78***	85.93	81.34	128.42	78.43
3/20	26.26	26.51	23.42	25.89	24.33	**17.82**	26.74	23.62	**16.01**	23.82	**15.10**	**9.63***	21.12
3/40	44.48	44.56	**34.57**	44.94	**40.14**	**28.79**	44.24	**38.83**	**25.50***	**36.96**	**27.62**	**26.22**	43.26
3/60	50.07	48.04	**36.57**	48.89	**44.87**	**31.46**	49.47	**41.61**	**29.25***	**40.78**	**33.29**	**35.48**	46.35
3/80	49.32	46.28	**39.98**	47.92	**43.37**	**35.25**	46.59	**41.64**	**33.68***	**41.80**	**38.41**	48.13	44.59

While the performance of the EMO part in the MOGLS algorithm was significantly improved by the hybridization, the improvement was not large in Table 3 and Table 4. One possible cause of the limited improvement is the negative effect of the selection mechanism in the local search part (i.e., the decrease in the variety of solutions). Thus we examined the performance of the hybrid SPEA and NSGA-II with no selection mechanism in the local search part as in Table 2. Average results are summarized in Table 5 and Table 6. From the comparison of Table 5 with Table 3, we can see that the performance of the hybrid SPEA for the three-objective problems was improved by removing the selection mechanism while the performance for the two-objective problems was deteriorated. This may be because the variety of solutions is more important in the three-objective problems with much more Pareto-optimal solutions. On the other hand, the performance of the hybrid NSGA-II was improved for both the two-objective and three-objective problems in many combinations of P_{LS} and k. Since the NSGA-II does not have a secondary population, the negative effect of the decrease in the variety of solutions may be more severe than the SPEA.

In the above computational experiments, we always used the scalar fitness function in the local search part. We also examined the use of the dominance relation based on the n objectives. In this case, a local move to a neighbor was accepted only when the current solution was dominated by the neighbor. As in Table 5 and Table 6 (i.e., without the selection mechanism in the local search part), we examined the performance of the hybrid SPEA and the hybrid NSGA-II with the acceptance criterion of a local move based on the dominance relation. Average results are summarized in Table 7 and Table 8. From Table 5 ~ Table 8, we can see that the use of the dominance relation as the acceptance criterion did not improve the performance of the hybrid algorithms in our computational experiments.

Table 5. Performance of the hybrid SPEA without the selection mechanism in the local search part.

Test Problem	$P_{LS} = 0.01$			$P_{LS} = 0.05$			$P_{LS} = 0.1$			$P_{LS} = 1$			SPEA
	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	
2/20	6.21	**5.93***	**5.99**	6.13	6.48	6.20	6.45	6.59	6.28	6.72	7.59	6.32	6.05
2/40	**17.13***	18.39	19.50	18.77	18.64	22.13	18.78	20.35	23.45	19.82	28.74	30.09	18.01
2/60	22.56	**21.26***	**21.75**	23.02	22.67	23.70	22.40	24.18	23.96	24.75	29.96	29.56	22.09
2/80	**81.04**	**76.07***	**76.41**	**76.12**	**76.29**	92.11	**79.39**	**81.12**	102.28	87.43	125.00	153.24	81.18
3/20	**6.93***	7.09	7.19	**6.98**	7.23	8.03	7.18	7.21	8.43	7.33	8.52	9.51	7.02
3/40	15.86	16.72	20.48	16.54	17.80	22.82	16.19	18.76	23.84	18.90	28.24	29.02	15.81*
3/60	**22.70***	**23.23**	26.32	**23.26**	24.42	29.50	24.17	26.77	30.26	26.98	38.94	38.92	24.10
3/80	**27.15**	**26.72***	32.12	28.20	28.71	36.09	28.44	30.82	38.86	31.79	46.65	53.20	28.10

Table 6. Performance of the hybrid NSGA-II without the selection mechanism in the local search part.

Test Problem	$P_{LS} = 0.01$			$P_{LS} = 0.05$			$P_{LS} = 0.1$			$P_{LS} = 1$			NSGA-II
	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	
2/20	9.37	**9.04**	**8.67**	**8.63**	**9.18**	**7.36**	**8.80**	**9.13**	**7.00**	**8.96**	**7.80**	**6.24***	9.25
2/40	22.84	**20.81**	**19.94***	**21.48**	**21.37**	**21.17**	21.72	**20.94**	21.77	**20.77**	26.85	30.76	21.54
2/60	**21.69**	**21.88**	**20.95***	**21.55**	**21.46**	22.26	**21.93**	22.39	23.10	**21.74**	27.62	30.76	22.04
2/80	**77.69**	**75.40**	**74.58**	**74.27***	**75.16**	90.20	83.14	81.51	98.04	85.18	128.85	163.79	78.43
3/20	22.84	21.55	**17.03**	**22.11**	**19.86**	**12.20**	21.39	**17.97**	**11.48**	**19.98**	**11.74**	**9.44***	21.12
3/40	44.08	**39.83**	**27.83**	43.48	**36.26**	**24.88**	45.21	**34.37**	**24.69***	**36.31**	**30.85**	**29.70**	43.26
3/60	**43.01**	**42.52**	**30.44**	**43.78**	**37.66**	**30.03***	**44.23**	**37.56**	**31.14**	**39.12**	**39.71**	**40.29**	46.35
3/80	**44.00**	**42.07**	**34.61***	**43.90**	**39.59**	**36.28**	**43.10**	**39.76**	**39.67**	**39.53**	46.91	54.32	44.59

Table 7. Performance of the hybrid SPEA without the selection mechanism in the local search part. The local move to a neighbor was accepted only when the current solution was dominated by the neighbor.

Test Problem	$P_{LS} = 0.01$			$P_{LS} = 0.05$			$P_{LS} = 0.1$			$P_{LS} = 1$			SPEA
	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	
2/20	**5.85***	**5.93**	6.60	6.06	6.07	7.87	6.18	6.95	8.37	7.08	8.60	10.59	6.05
2/40	**17.94**	18.04	21.39	18.74	19.85	28.64	**17.65***	19.82	31.09	21.14	31.33	46.01	18.01
2/60	22.10	23.24	25.24	22.69	24.30	28.14	22.12	24.83	30.32	25.43	31.73	42.11	22.09*
2/80	**79.90***	82.41	97.35	82.27	88.03	135.08	**80.86**	96.36	164.78	96.51	144.53	316.44	81.18
3/20	7.15	7.18	7.93	7.06	7.32	9.97	7.06	7.69	10.99	7.71	10.12	14.39	7.02*
3/40	16.27	17.20	19.87	16.35	18.57	26.82	16.28	18.30	29.10	19.50	28.61	39.31	15.81*
3/60	**22.96***	24.73	28.44	**23.41**	25.12	34.04	**23.61**	27.40	37.94	28.09	38.87	50.61	24.10
3/80	**27.40**	**27.26***	33.19	**27.65**	29.45	40.37	28.27	32.33	45.38	32.33	46.12	65.85	28.10

Table 8. Performance of the hybrid NSGA-II without the selection mechanism in the local search part. The local move to a neighbor was accepted only when the current solution was dominated by the neighbor.

Test Problem	$P_{LS} = 0.01$			$P_{LS} = 0.05$			$P_{LS} = 0.1$			$P_{LS} = 1$			NSGA-II
	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	$k = 1$	10	100	
2/20	9.52	9.30	**8.64**	**8.80**	**8.52**	**8.45**	9.38	**9.00**	**8.17***	**8.98**	**9.16**	10.99	9.25
2/40	21.74	**21.53**	**21.18**	**20.08***	**20.77**	26.45	**21.06**	22.45	28.76	**21.37**	27.90	48.46	21.54
2/60	**21.63**	**21.84**	23.28	**22.02**	22.21	27.53	**21.50***	22.41	30.82	23.35	29.59	43.10	22.04
2/80	82.14	82.02	97.92	**75.19***	83.71	150.30	82.15	95.09	194.08	92.72	152.54	322.61	78.43
3/20	22.44	21.30	**20.11**	22.45	**20.20**	**15.71**	22.08	**19.92**	**13.86**	**20.71**	**14.37**	**13.73***	21.12
3/40	44.99	**42.99**	**30.49**	**43.08**	**37.07**	**27.56***	**42.52**	**35.37**	**29.34**	**35.60**	**29.99**	**40.49**	43.26
3/60	**45.05**	**42.41**	**33.75**	**45.57**	**40.07**	**33.71***	**44.07**	**38.00**	**36.90**	**39.76**	**39.90**	52.33	46.35
3/80	**43.59**	**44.28**	**36.89***	**44.01**	**42.13**	**40.87**	**44.36**	**40.95**	44.64	**41.13**	45.96	67.95	44.59

4. Conclusion

In this paper, we examined how the performance of EMO algorithms can be improved by the hybridization with local search. We showed through computational experiments on multi-objective permutation flowshop scheduling problems that the performance of the EMO part of our MOGLS algorithm was improved by the hybridization with local search while the performance of the MOGLS algorithm strongly depended on the choice of the parameter values in the local search part (i.e., k and P_{LS}). We also implemented hybrid versions of the SPEA and the NSGA-II. The hybridization with local search improved the performance of the SPEA and the NSGA-II when k and P_{LS} were appropriately specified. The best results for the two-objective test problems were obtained by our MOGLS algorithm and the hybrid SPEA with the selection mechanism of initial solutions in the local search part. On the other hand, the best results for the three-objective test problems were obtained by the non-hybrid SPEA and the hybrid SPEA with no selection mechanism of initial solutions in the local search part. One potential advantage of hybrid algorithms over pure EMO algorithms is the decrease in the CPU time. This is because local search can be much more efficiently executed than genetic search in many application problems.

This study is partially supported by Japan Society for the Promotion of Science (JSPS) through Grand-in-Aid for Scientific Research (B): KAKENHI (14380194).

References

[1] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," *Proc. of 1st International Conference on Genetic Algorithms and Their Applications*, pp. 93-100, 1985.

[2] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons, 2001.

[3] C. A. Coello Coello, D. A. van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*, Kluwer Academic Publishers, 2002.

[4] P. Moscato, "Memetic algorithms: A short introduction," in D. Corne, F. Glover, and M. Dorigo (eds.), *New Ideas in Optimization*, McGraw-Hill, pp. 219-234, Maidenhead, 1999.

[5] Memetic Algorithms' Home Page: http://www.densis.fee.unicamp.br/~moscato/memetic_home.html.

[6] H. Ishibuchi and T. Murata, "Multi-objective genetic local search algorithm," *Proc. of 3rd IEEE International Conference on Evolutionary Computation*, pp. 119-124, 1996.

[7] H. Ishibuchi and T. Murata, "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*, vol. 28, no. 3, pp. 392-403, 1998.

[8] A. Jaszkiewicz, "Genetic local search for multi-objective combinatorial optimization," *European Journal of Operational Research*, vol. 137, no. 1, pp. 50-71, 2002.

[9] H. Ishibuchi, T. Yoshida, and T. Murata, "Selection of initial solutions for local search in multiobjective evolutionary algorithms," *Proc. of 2002 Congress on Evolutionary Computation*, pp. 950-955, 2002.

[10] H. Ishibuchi and T. Yoshida, and T. Murata, "Balance between genetic search and local search in hybrid evolutionary multi-criterion optimization algorithms," *Proc. of 2002 Genetic and Evolutionary Computation Conference*, pp. 1301-1308, 2002.

[11] J. D. Knowles and D. W. Corne, "M-PAES: A memetic algorithm for multiobjective optimization," *Proc. of 2000 Congress on Evolutionary Computation*, pp. 325-332, 2000.

[12] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using Pareto archived evolution strategy," *Evolutionary Computation*, vol. 8, no. 2, pp. 149-172, 2000.

[13] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. on Evolutionary Computation*, vol. 3, no. 4, pp. 257-271, 1999.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

[15] T. Murata and H. Ishibuchi, "MOGA: Multi-objective genetic algorithms," *Proc. of 1995 IEEE International Conference on Evolutionary Computation*, pp.289-294, 1995.