Effects of Using Two Neighborhood Structures on the Performance of Cellular Evolutionary Algorithms for Many-Objective Optimization

Hisao Ishibuchi, Yuji Sakane, Noritaka Tsukamoto, and Yusuke Nojima

Abstract— **Cellular evolutionary algorithms usually use a single neighborhood structure for local selection. When a new solution is to be generated by crossover and/or mutation for a cell, a pair of parent solutions is selected from its neighbors. The current solution at the cell is replaced with the newly generated offspring if the offspring has the higher fitness value than the current one. That is, the "replace-if-better" policy is used for the replacement of the current solution. Local selection, crossover, mutation and replacement are iterated at every cell in cellular algorithms. A recently proposed multiobjective evolutionary algorithm called MOEA/D by Zhang and Li (2007) can be viewed as a cellular algorithm where each cell has its own scalarizing fitness function with a different weight vector. We can introduce a spatial structure to MOEA/D by the Euclidean distance between weight vectors. Its main difference from standard cellular algorithms is that a newly generated offspring for a cell is compared with not only the current solution of the cell but also its neighbors for local replacement in MOEA/D. In this paper, we examine the effect of local replacement on the search ability of a cellular version of MOEA/D. Whereas the same neighborhood structure was used for local selection and local replacement in the original MOEA/D, we examine the use of different neighborhood structures for local selection and local replacement. It is shown through computational experiments on multiobjective 0/1 knapsack problems with two, four and six objectives that local replacement plays an important role in MOEA/D especially for many-objective optimization problems.**

I. Introduction

Currently the handling of many-objective problems is a hot issue in the EMO (evolutionary multiobjective optimization) community [1], [2]. Whereas a large number of EMO algorithms have been proposed and successfully used in a wide range of real-world application tasks [3]-[7], Pareto dominance-based EMO algorithms such as SPEA [8] and NSGA-II [9] do not work well on many-objective problems with four or more objectives [10]-[14]. This is because almost all solutions in each population become non-dominated when they are compared with each other with respect to many objectives. As a result, enough selection pressure toward the Pareto front cannot be generated by the Pareto dominance relation. Thus we may need different fitness evaluation mechanisms for many-objective problems instead of the Pareto dominance relation.

One promising fitness evaluation approach, which is not based on the Pareto dominance relation between solutions, is the use of an indicator function for measuring the quality of solution sets [15]-[24]. The fitness of each solution is defined by its contribution to the indicator function. EMO algorithms with indicator functions are referred to as indicator-based evolutionary algorithms (IBEAs). The hypervolume [25] has often been used in IBEAs since it has nice features as a quality measure of solution sets [26]-[28]. High search ability and high flexibility of IBEAs have been reported in the literature (e.g., see [19] for their search ability on various test problems and [18], [20] for their flexibility to incorporate the decision maker's preference). IBEAs have a number of good theoretical features (e.g., their direct optimization ability of non-dominated solution sets as single-objective optimizers and the Pareto-compliant property of the hypervolume). Their main difficulty is the exponential increase in the computation load for hypervolume calculation as the number of objectives increases. Thus the applications of IBEAs to many-objective problems with more than six objectives are very difficult.

Another promising fitness evaluation approach is the use of scalarizing fitness functions such as the weighted sum, the Tchebycheff function, and the achievement scalarizing function [12]-[14], [29]-[32]. The use of scalarizing fitness functions has not been popular in the EMO community. They were often used for local search in memetic EMO algorithms [12], [33]-[35]. The main advantage of scalarizing fitness functions is the simplicity of their calculation. The computational load for fitness evaluation does not increase exponentially with the number of objectives. Thus scalarizing fitness functions usually need much less computation load for fitness evaluation than the hypervolume calculation.

Recently Zhang and Li [32] proposed a multiobjective evolutionary algorithm based on decomposition (MOEA/D), which is a scalarizing function-based EMO algorithm. This is a powerful and simple EMO algorithm. That is, it has high search ability for continuous optimization, combinatorial optimization, and many-objective optimization. Moreover, it does not use any complicated fitness evaluation mechanisms. Multiobjective evolution is driven only by scalarizing fitness functions. MOEA/D uses a set of uniformly distributed weight vectors. Solutions are evaluated by scalarizing fitness functions with different weight vectors. Each scalarizing fitness function (i.e., each weight vector) keeps a single solution which is the best with respect to the scalarizing fitness function with its own weight vector. When a new

Manuscript received November 14, 2008. This work was supported in part by the Japan Society for the Promotion of Science under Grant-in-Aid for Scientific Research (B) (20300084).

Hisao Ishibuchi, Yuji Sakane, Noritaka Tsukamoto, and Yusuke Nojima are with the Department of Computer Science and Intelligent Systems, Osaka Prefecture University, 1-1 Gakuen-cho, Naka-ku, Sakai, Osaka 599-8531, Japan (hisaoi@cs.osakafu-u.ac.jp, sakane@ci.cs.osakafu-u.ac.jp, nori@ci.cs.osakafu-u.ac.jp, nojima@cs.osakafu-u.ac.jp).

978-1-4244-2959-2/09/$25.00 ©

offspring is to be generated for a scalarizing fitness function, a pair of solutions is randomly selected as parents from its neighbors (including the current solution of the scalarizing function). Neighbors are defined by the Euclidean distance between weight vectors. The newly generated offspring from the selected pair of parent solutions is compared with each neighbor using the neighbor's scalarizing fitness function. If the offspring is better than a neighbor, the neighbor is replaced with the offspring. The comparison for replacement is performed for all neighbors (including the current solution for which the offspring is generated). During the execution of MOEA/D, all the non-dominated solutions among the examined ones are stored in an archive population. The archive population is used only to store the non-dominated solutions (i.e., no solutions in the archive population are used as parents to generate a new offspring). Thus the archive population has no effect on the behavior of MOEA/D while it has a large effect on the finally obtained solution set.

From these explanations, one may think that MOEA/D can be viewed as a cellular algorithm. Whereas there was no description on its cellular implementation in the original paper of MOEA/D [32], we handle it as a cellular algorithm in this paper. The main characteristic feature of MOEA/D as a cellular algorithm is the use of local replacement in addition to local selection. That is, a newly generated offspring for a cell is compared with not only the current solution of the cell but also its neighbors for possible replacement. In standard cellular algorithms, a newly generated offspring for a cell is usually compared with only the current solution of the cell.

This paper is organized as follows. In Section II, we explain MOEA/D with no archive population as a cellular algorithm. We implement such a cellular version of MOEA/D. While the same neighborhood structure was used for local selection and local replacement in the original MOEA/D, our cellular MOEA/D has different neighborhood structures for local selection and local replacement. In Section III, we examine the performance of our cellular MOEA/D through computational experiments on multiobjective 0/1 knapsack problems with two, four and six objectives. We examine various specifications of the two neighborhood structures and the population size in our computational experiments. We also compare our cellular MOEA/D with NSGA-II. Finally we conclude this paper in Section IV.

II. Cellular Version of MOEA/D

A. Multiobjective Optimization Problem

A k-objective maximization problem is written as follows:

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

$$\text{subject to } \mathbf{x} \in \mathbf{X}, \quad (2)$$

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

Let $\mathbf{y}$ and $\mathbf{z}$ be two feasible solutions of the k-objective maximization problem in (1)-(2). If the following conditions hold, $\mathbf{z}$ can be viewed as being better than $\mathbf{y}$:

$$\forall i : f_i(\mathbf{y}) \le f_i(\mathbf{z}) \quad \text{and} \quad \exists j : f_j(\mathbf{y}) < f_j(\mathbf{z}). \quad (3)$$

In this case, we say that $\mathbf{z}$ dominates $\mathbf{y}$ (equivalently $\mathbf{y}$ is dominated by $\mathbf{z}$: $\mathbf{z}$ is better than $\mathbf{y}$).

When $\mathbf{y}$ is not dominated by any other feasible solutions, $\mathbf{y}$ is referred to as a Pareto-optimal solution of the k-objective maximization problem in (1)-(2). The set of all Pareto-optimal solutions forms the tradeoff surface in the objective space. This tradeoff surface is referred to as the Pareto front. EMO algorithms are usually designed to search for a set of well-distributed non-dominated solutions that approximates the entire Pareto front very well.

B. Scalarizing Fitness Functions

The weighted sum and the Tchebycheff function were used as scalarizing fitness functions in [32]. We also examine the use of these scalarizing fitness functions in our computational experiments. The weighted sum of the k objectives in (1) can be written for a weight vector $\mathbf{w} = (w_1, ..., w_k)$ as follows:

$$g_{\mathrm{W}}(\mathbf{x}, \mathbf{w}) = w_1 \cdot f_1(\mathbf{x}) + \cdots + w_k \cdot f_k(\mathbf{x}), \quad (4)$$

where the weight vector $\mathbf{w}$ satisfies the following conditions:

$$w_1 + \cdots + w_k = 1 \quad \text{and} \quad w_i \ge 0 \text{ for } i = 1, 2, ..., k. \quad (5)$$

The Tchebycheff function needs a reference point $\mathbf{z}^*$ in the objective space: $\mathbf{z}^* = (z_1^*, ..., z_k^*)$. Let Ω be a set of solutions. The reference point can be specified by the best objective value of each objective in Ω as

$$z_i^* = \max\{f_i(\mathbf{x}) \mid \mathbf{x} \in \Omega\}, \ i = 1, 2, ..., k. \quad (6)$$

When Ω is the set of all feasible solutions, $\mathbf{z}^*$ is called the ideal vector.

The Tchebycheff function can be written for a weight vector $\mathbf{w} = (w_1, ..., w_k)$ as follows:

$$g_{\mathrm{T}}(\mathbf{x}, \mathbf{w}, \mathbf{z}^*) = \max_{i=1,2,...,k} \{w_i \cdot | z_i^* - f_i(\mathbf{x}) |\}. \quad (7)$$

As in the original MOEA/D of Zhang and Li [32], we specify the reference point $\mathbf{z}^*$ for each generation as follows in our computational experiments:

$$z_i^* = 1.1 \cdot \max\{f_i(\mathbf{x}) \mid \mathbf{x} \in \Omega(t)\}, \ i = 1, 2, ..., k, \quad (8)$$

where $\Omega(t)$ is the current population at the t-th generation. This means that the reference point $\mathbf{z}^*$ is updated at every generation.

C. Weight Vector Specification

First, uniformly distributed weight vectors are specified under the normalization condition (5) in MOEA/D. Each weight vector can be viewed as a point in the k-dimensional unit hypercube $[0, 1]^k$. We use such a point as a cell in our cellular implementation of MOEA/D. Thus the number of

cells (i.e., the population size) is the same as the number of weight vectors. As in [32], we generate all weight vectors satisfying the following conditions:

$$w_1 + w_2 + \cdots + w_k = 1, \tag{9}$$

$$w_i \in \left\{0, \frac{1}{H}, \frac{2}{H}, ..., \frac{H}{H}\right\}, \ i = 1, 2, ..., k, \tag{10}$$

where H is a user-definable positive integer, which can be viewed as the granularity of weight value discretization. For example, we have 101 weight values 0, 0.01, ..., 1.00 in (10) when H is specified as H=100. In the case of two objectives, this specification leads to 101 weight vectors (0, 1), (0.01, 0.99), ..., (1, 0). Almost the same specification method was used in Murata et al. [36] for their cellular EMO algorithm.

D. Cellular MOEA/D Algorithm

As we have already mentioned in the previous subsection, each cell has its own weight vector. The number of cells is the same as the number of different weight vectors. The location of each cell in the k-dimensional unit hypercube is the point specified by its weight vector. Thus the cells are uniformly located in the $(k-1)$-dimensional subspace satisfying the normalization condition in (5) in the k-dimensional unit hypercube. For example, they are uniformly located on the line between (1, 0) and (0, 1) in the unit square in the case of two-objective problems. For three-objective problems, they are uniformly located on the triangular plane with the three vertices (1, 0, 0), (0, 1, 0) and (0, 0, 1) in the unit cube.

We implement MOEA/D as a cellular algorithm with two neighborhood structures: One is for local selection and the other is for local replacement. We define them using the Euclidean distance between cells (i.e., between their weight vectors). More specifically, the neighbors of a cell are defined as a prespecified number of the nearest neighbors (including that cell). In this manner, we can examine any neighborhood structure of an arbitrarily specified neighborhood size.

An initial solution is randomly generated for each cell as in standard cellular algorithms. If the generated solution is infeasible, we use a repair procedure for obtaining a feasible solution from the generated infeasible one. The fitness of the solution is calculated at each cell using a scalarizing fitness function with the weight vector associated with that cell.

In order to generate an offspring for each cell, a pair of parent solutions is randomly selected from its selection neighbors. A feasible offspring is generated from the selected pair of parent solutions by crossover, mutation and/or repair. The generated offspring is compared with each of its replacement neighbors. When the offspring is compared with a neighbor, the neighbor's weight vector is used. If the offspring is better than the neighbor, the neighbor is replaced with the offspring. It should be noted that we execute this replacement procedure sequentially. That is, local selection and local replacement are performed for the next cell after those procedures are completed for the current cell.

Our cellular MOEA/D can be written as follows:

Parameter Specifications: Specify the value of H in (9)-(10), the number of selection neighbors, the number of replacement neighbors, the crossover probability, the mutation probability, and the termination condition.

Initialization: Generate weight vectors by (9)-(10). A single weight vector is assigned to each cell. Generate an initial solution for each cell.

Multiobjective Evolution: Apply the following steps to each cell sequentially and iteratively until the termination condition is satisfied:

(1) Select a pair of parent solutions from the selection neighbors of the current cell.

(2) Generate an offspring from the selected pair of parent solutions by crossover, mutation and/or repair.

(3) Compare the generated offspring with each of the replacement neighbors of the current cell. If the offspring is better, replace the neighbor with the offspring.

III. Computational Experiments

A. Settings of Computational Experiments

In our computational experiments, we used multiobjective 0/1 knapsack problems with two, four and six objectives and 500 items. We denote these problems as 2-500, 4-500 and 6-500. The 2-500 and 4-500 knapsack problems are the same as in Zitzler and Thiele [8]. We generated the 6-500 knapsack problem in the same manner as the other test problems in [8]. Solutions of these problems were coded as binary strings of length 500. We used uniform crossover and bit-flip mutation. Infeasible solutions were repaired by removing items in a greedy manner as in [32]. This greedy repair chose items using a scalarizing fitness function (for detail, see [32]).

We used the weighted sum and the Tchebycheff function as a scalarizing fitness function in our cellular MOEA/D. As we have already mentioned, we used no archive population. Thus the final solution set of our cellular MOEA/D was non-dominated solutions in the final population of each run. We report average results over 30 runs in this paper.

We used the hypervolume [25] as an overall performance measure to evaluate non-dominated solution sets. In order to intuitively examine the convergence and diversity properties of our cellular MOEA/D, we also used two simple measures in addition to the hypervolume. Of course, other performance measures are available in the literature (e.g., see [26]-[28]).

One measure MaxSum is the maximum value of the sum of all objectives:

$$\text{MaxSum}(\Omega) = \max_{\mathbf{x} \in \Omega} \sum_{i=1}^{k} f_i(\mathbf{x}), \tag{11}$$

where Ω denotes the final population. MaxSum is an intuitive measure to evaluate the convergence of solutions toward the center region of the Pareto front. The other measure Range is the sum of the ranges of objective values over all objectives:

$$\text{Range}(\Omega) = \sum_{i=1}^{k} [\max_{\mathbf{x} \in \Omega} \{f_i(\mathbf{x})\} - \min_{\mathbf{x} \in \Omega} \{f_i(\mathbf{x})\}]. \tag{12}$$

Range is a simple measure for the diversity of solutions.

For comparison, we also applied NSGA-II [9] to the 2-500, 4-500 and 6-500 test problems. We used the same genetic operators in both our cellular MOEA/D and NSGA-II (i.e., uniform crossover and bit-flip mutation). Since NSGA-II had no scalarizing function, we used a greedy repair method based on the maximum profit/weight ratio in NSGA-II as in Zitzler and Thiele [8].

B. Effects of Two Neighborhood Structures

We applied our cellular MOEA/D to the three test problems using the following parameter specifications:

Crossover probability: 1.0 (uniform crossover),
Mutation probability: 0.002 per bit (bit-flip mutation),
Number of selection neighbors: 2, 5, 10, 20, 50,
Number of replacement neighbors: 1, 2, 5, 10, 20, 50.

We examined all the 5x6 combinations of the different specifications for selection neighborhood and replacement neighborhood. The other parameter values are summarized in Table I where the number of examined solutions in the last column is the termination condition for each test problem.

For comparison, we applied NSGA-II to each test problem using the same population size and termination condition as those in Table I. We examined various combinations of the crossover probability and the mutation probability as follows:

Crossover probability: 0.4, 0.6, 0.8, 1.0,
Mutation probability: 0.001, 0.002, 0.004, 0.008.

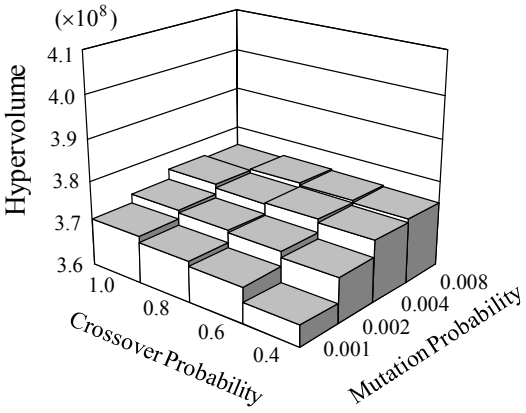
In Fig. 1, we show experimental results by NSGA-II. We can see from this figure that the performance of NSGA-II was not so sensitive to the specifications of the crossover and mutation probabilities.

Experimental results by our cellular MOEA/D are shown in Fig. 2 and Fig. 3. The Tchebycheff function was used in Fig. 2 while the weighted sum fitness function was used in Fig. 3. Almost the same results were obtained for the 2-500 problem from these two scalarizing fitness functions in Fig. 2 (a) and Fig. 3 (a). For many-objective problems, better results were obtained from the weighted sum fitness function in Fig. 3 than the Tchebycheff function in Fig. 2. Since the contour lines of the Tchebycheff function have the same shape as the boundary of the region dominating a point in the objective space, the Pareto dominance relation and the Tchebycheff function have similar search behaviors in multiobjective optimization. That is, their search ability is severely degraded by the increase in the number of objectives.

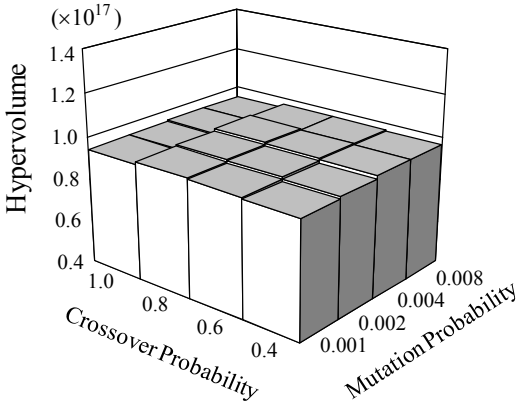
From the comparison of Fig. 1 by NSGA-II with Fig. 2 and Fig. 3 by our cellular MOEA/D, we can see that much better results were obtained by our cellular MOEA/D. The large difference in the hypervolume between the two algorithms in Figs. 1-3 is partially because they used the different fitness evaluation mechanisms. It is, however, mainly because they used the different repair schemes. Actually the choice of a repair scheme has a large effect on the performance of EMO algorithms for multiobjective 0/1 knapsack problems [37].

TABLE I
PARAMETER SPECIFICATIONS IN SUBSECTION III.B

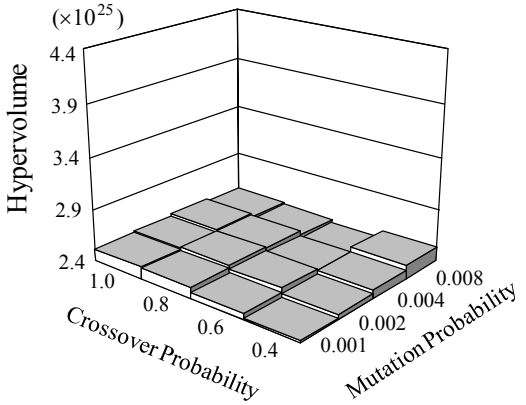
Test problems	Value of H	Population size: Weight vectors	Examined solutions
2-500	199	200	100,000
4-500	12	455	150,000
6-500	6	462	200,000



(a) Experimental results on the 2-500 test problem by NSGA-II.



(b) Experimental results on the 4-500 test problem by NSGA-II.

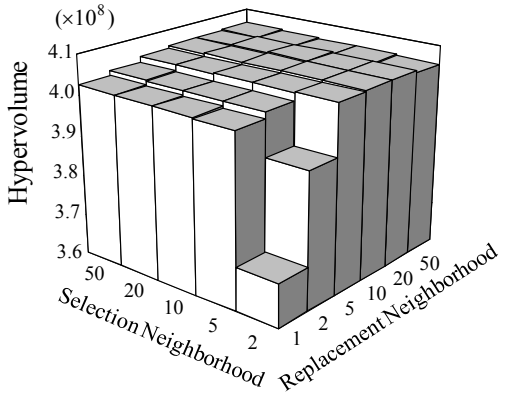


(c) Experimental results on the 6-500 test problem by NSGA-II.

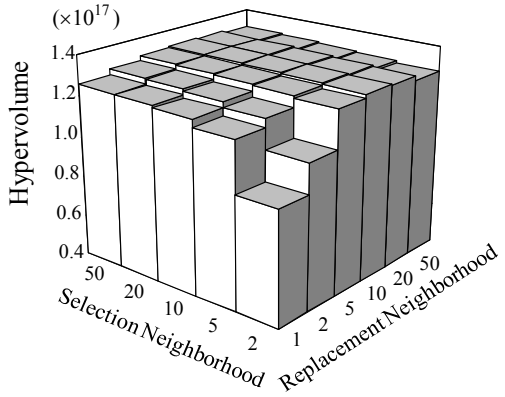
Fig. 1. Experimental results by NSGA-II (Hypervolume).

Let us examine the experimental results by our cellular MOEA/D in Fig. 2 and Fig. 3 in detail. In each plot of these figures, the leftmost row shows the results with a single replacement neighbor. This specification corresponds to standard cellular algorithms where a new offspring for a cell is compared only with the current solution in that cell. In almost all plots, better results were obtained by increasing the number of replacement neighbors.

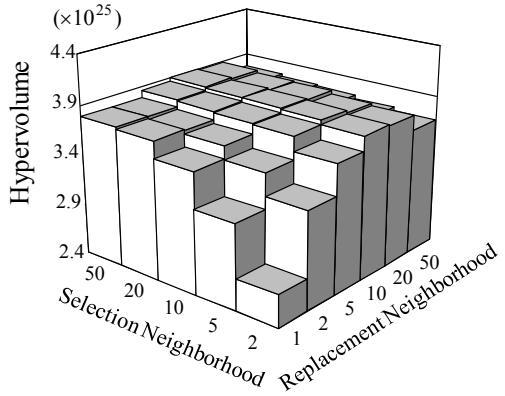
This improvement is more prominent in the case of many objectives than two objectives. In all plots in Fig. 2 and Fig. 3, good results were not obtained from the combination of two selection neighbors and a single competition neighbor. This is because the propagation of good solutions from the current cells to other cells was very slow in this combination. By simply increasing the number of replacement neighbors, the performance of our cellular MOEA/D was clearly improved.



(a) Results on the 2-500 test problem by the Tchebycheff function.

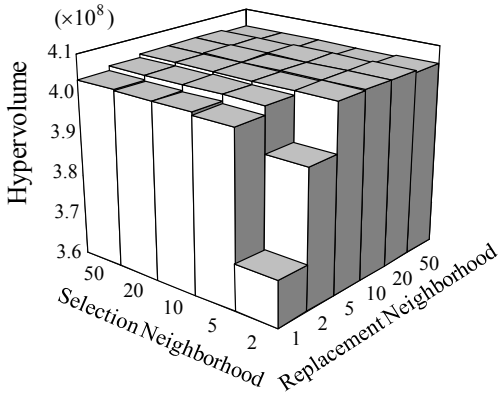


(b) Results on the 4-500 test problem by the Tchebycheff function.

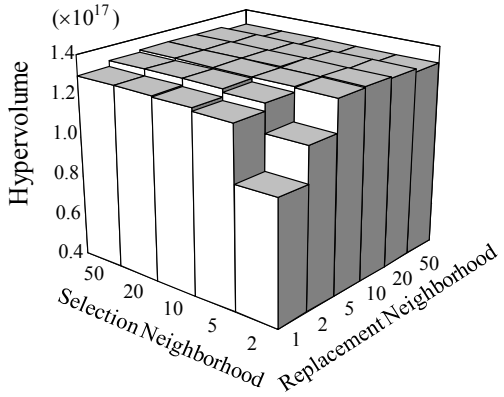


(c) Results on the 6-500 test problem by the Tchebycheff function.

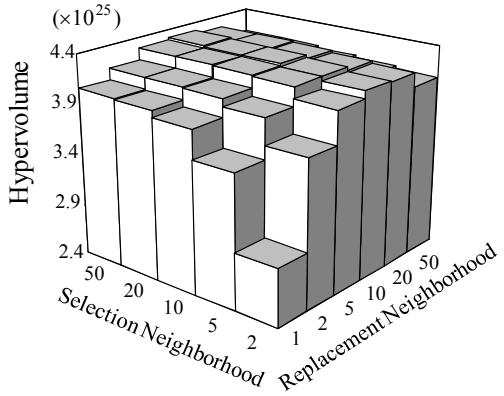
Fig. 2. Hypervolume by cellular MOEA/D with the Tchebycheff function.



(a) Results on the 2-500 test problem by the weighted sum fitness function.



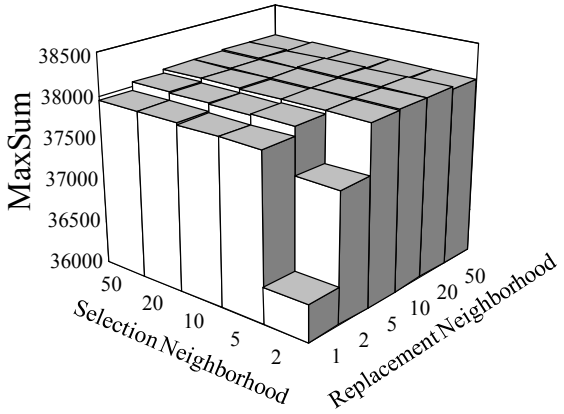
(b) Results on the 4-500 test problem by the weighted sum fitness function.



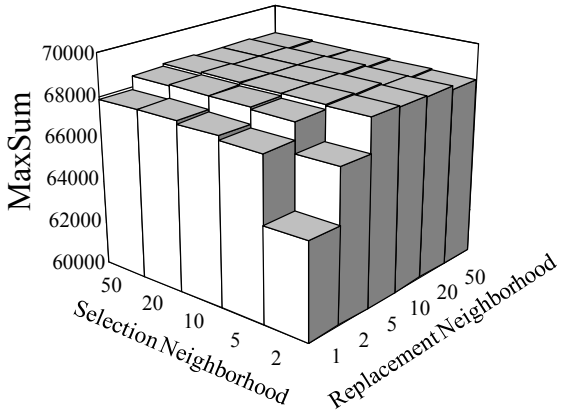
(c) Results on the 6-500 test problem by the weighted sum fitness function.

Fig. 3. Hypervolume by cellular MOEA/D with the weighted sum.

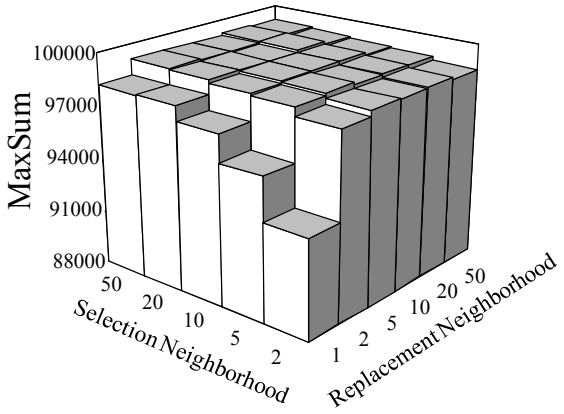
In Fig. 4, we show experimental results with respect to the MaxSum measure by our cellular MOEA/D with the weighted sum fitness function. Each plot in Fig. 4 shows the convergence performance of our cellular MOEA/D. From Fig. 4, we can see that the convergence performance was improved by increasing the number of replacement neighbors. We can also see that the convergence performance in Fig. 4 is very similar to the overall performance in Fig. 3.



(a) Results on the 2-500 test problem by the weighted sum fitness function.



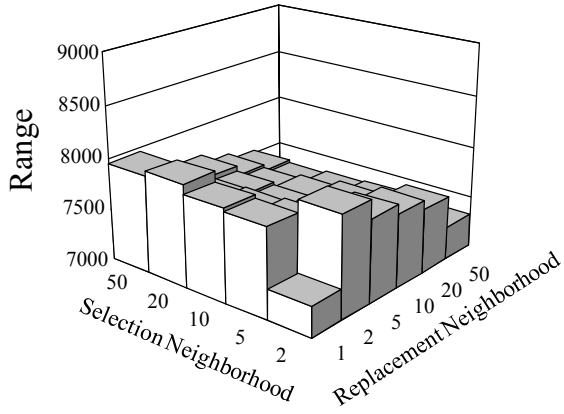
(b) Results on the 4-500 test problem by the weighted sum fitness function.



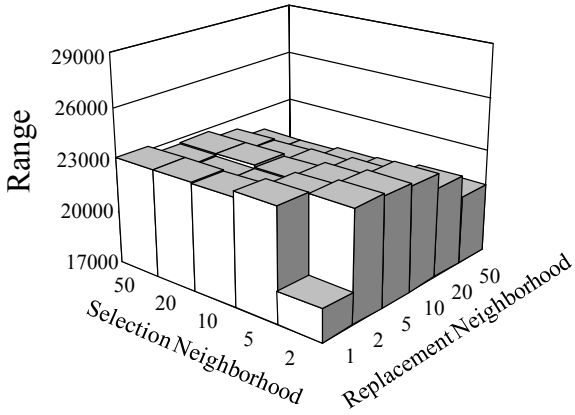
(c) Results on the 6-500 test problem by the weighted sum fitness function.

Fig. 4. MaxSum by our cellular MOEA/D with the weighted sum.

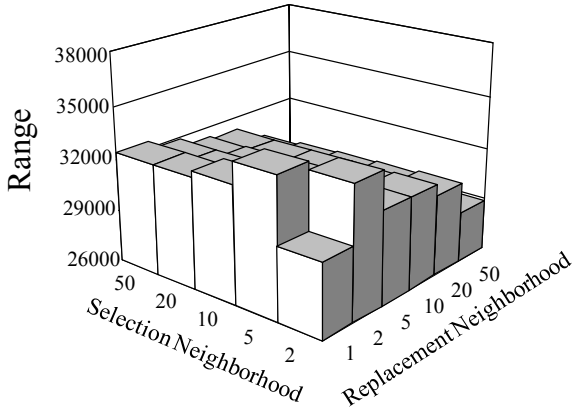
In Fig. 5, we show experimental results with respect to the Range measure by our cellular MOEA/D algorithm with the weighted sum fitness function. Since much better results were obtained by the weighted sum in Fig. 3 than the Tchebycheff function in Fig. 2, we only report experimental results by the weighted sum in Fig. 4 and Fig. 5. From Fig. 5, we can see that the increase in the number of replacement neighbors clearly decreases the diversity of solutions.



(a) Results on the 2-500 test problem by the weighted sum fitness function.



(b) Results on the 4-500 test problem by the weighted sum fitness function.



(c) Results on the 6-500 test problem by the weighted sum fitness function.

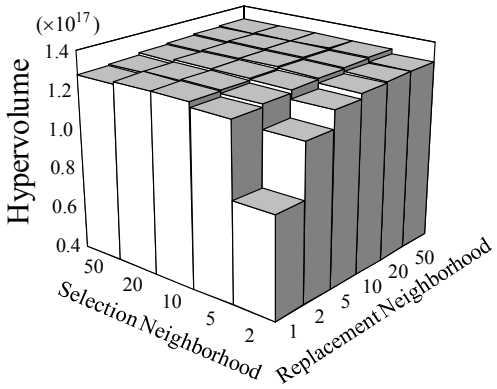
Fig. 5. Range by our cellular MOEA/D with the weighted sum.

C. Effects of Population Size

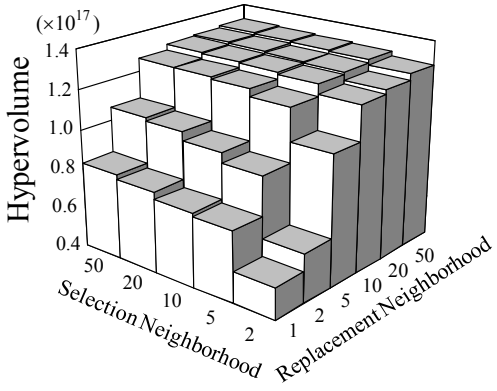
We examined the effect of the population size on the performance of our cellular MOEA/D through computational experiments on the 4-500 problem under the conditions:

Crossover probability: 1.0 (uniform crossover),
Mutation probability: 0.002 per bit (bit-flip mutation),
Population size: 969 ($H=16$) and 9,880 ($H=37$),
Termination condition: 150,000 examined solutions,
Number of selection neighbors: 2, 5, 10, 20, 50,
Number of replacement neighbors: 1, 2, 5, 10, 20, 50.

Experimental results are summarized for the hypervolume in Fig. 6 and Fig. 7. From the comparison between Fig. 2 (b) with the population size 455 and Fig. 6 with the population size 969 and 9,880 (and the comparison between Fig. 3 (b) and Fig. 7), we can see that the performance of our cellular MOEA/D was improved by increasing the population size. We can also see from Fig. 6 (b) and Fig. 7 (b) that the number of the replacement neighbors has a large effect on the performance of our cellular MOEA/D. When the population size is large, we need to use a large number of replacement neighbors for obtained good results by our cellular MOEA/D (see Fig. 6 (b) and Fig. 7 (b)).

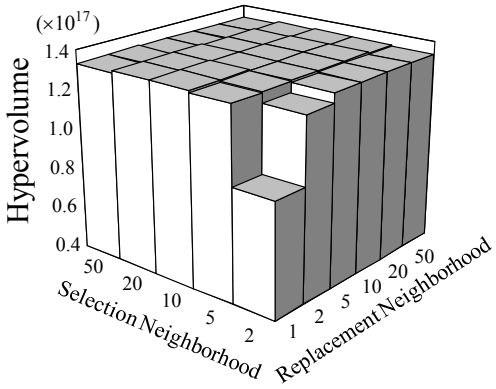


(a) Results on the 4-500 test problem where the population size was 969.

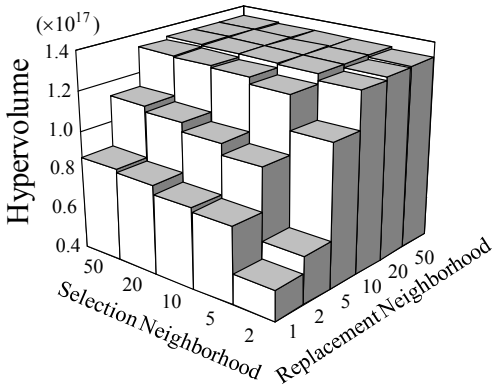


(b) Results on the 4-500 test problem where the population size was 9,880.

Fig. 6. Hypervolume by our cellular MOEA/D with Tchebycheff function.



(a) Results on the 4-500 test problem where the population size was 969.



(b) Results on the 4-500 test problem where the population size was 9,880.

Fig. 7. Hypervolume by our cellular MOEA/D with the weighted sum.

An interesting observation is that good results were not obtained when the number of replacement neighbors was one as in standard cellular algorithms in Fig. 6 (b) and Fig. 7 (b). Since the population size was 9,880 in those plots, the execution of our cellular MOEA/D was terminated at the 16th generation because the termination condition was 150,000 examined solutions. Good results were still obtained under such a strange parameter specification when the number of replacement neighbors was large in our cellular MOEA/D.

IV. Conclusion

We examined the effect of two neighborhood structures on the performance of a cellular version of MOEA/D, which outperformed NSGA-II on knapsack problems with two, four and six objectives. Experimental results showed that local replacement neighborhood had much larger effects on the performance of our cellular MOEA/D than local selection neighborhood. In almost all cases, the hypervolume measure was improved by increasing the size of local replacement neighborhood. This observation suggests the possibility to improve the performance of standard cellular algorithms by introducing the concept of local replacement. This is because standard cellular algorithms have only a single replacement neighbor (i.e., an offspring for a cell is compared with only

the current solution at that cell). We also demonstrated that good results were obtained even when the population size was very large (e.g., 9880). In this case, we need to use a large replacement neighborhood with at least 10 replacement neighbors.

REFERENCES

[1] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization: A short review," *Proc. of 2008 IEEE Congress on Evolutionary Computation*, pp. 2424-2431, Hong Kong, June 1-6, 2008.

[2] H. Ishibuchi, N. Tsukamoto, Y. Hitotsuyanagi, and Y. Nojima, "Effectiveness of scalability improvement attempts on the performance of NSGA-II for many-objective problems," *Proc. of 2008 Genetic and Evolutionary Computation Conference*, pp. 649-656, Atlanta, July 12-16, 2008.

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

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

[5] C. A. C. Coello and G. B. Lamont, *Applications of Multi-Objective Evolutionary Algorithms*, World Scientific, Singapore, 2004.

[6] A. Abraham, L. C. Jain, and R. Goldberg (eds.), *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications*, Springer, Berlin, 2005.

[7] K. C. Tan, E. F. Khor, and T. H. Lee, *Multiobjective Evolutionary Algorithms and Applications*, Springer, Berlin, 2005.

[8] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. on Evolutionary Computation* 3 (4), pp. 257-271, November 1999.

[9] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. on Evolutionary Computation* 6 (2), pp. 182-197, April 2002.

[10] V. Khara, X. Yao, and K. Deb, "Performance scaling of multi-objective evolutionary algorithms," *Lecture Notes in Computer Science 2632: Evolutionary Multi-Criterion Optimization - EMO 2003*, pp. 367-390, Springer, Berlin, April 2003.

[11] R. C. Purshouse and P. J. Fleming, "Evolutionary many-objective optimization: An exploratory analysis," *Proc. of 2003 IEEE Congress on Evolutionary Computation*, pp. 2066-2073, Canberra, December 8-12, 2003.

[12] A. Jaszkiewicz, "On the computational efficiency of multiple objective metaheuristics: The knapsack problem case study," *European Journal of Operational Research* 158 (2), pp. 418-433, October 2004.

[13] E. J. Hughes, "Evolutionary many-objective optimization: Many once or one many?," *Proc. of 2005 IEEE Congress on Evolutionary Computation*, pp. 222-227, Edinburgh, September 2-5, 2005.

[14] H. Ishibuchi, Y. Nojima, and T. Doi, "Comparison between single-objective and multi-objective genetic algorithms: Performance comparison and performance measures," *Proc. of 2006 IEEE Congress on Evolutionary Computation*, pp. 3959-3966, Vancouver, July 16-21, 2006.

[15] J. D. Knowles, D. W. Corne, and M. Fleischer, "Bounded archiving using the Lebesgue measure," *Proc. of 2003 IEEE Congress on Evolutionary Computation*, pp. 2490-2497, Canberra, December 8-12, 2003.

[16] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," *Lecture Notes in Computer Science 3242: Parallel Problem Solving from Nature - PPSN VIII*, pp. 832-842, Springer, Berlin, 2004.

[17] M. Emmerich, N. Beume, and B. Naujoks, "An EMO algorithm using the hypervolume measure as selection criterion," *Lecture Notes in Computer Science 3410: Evolutionary Multi-Criterion Optimization - EMO 2005*, pp. 62-76, Springer, Berlin, 2005.

[18] L. Thiele, K. Miettinen, P. J. Korhonen, and J. Molina, "A preference-based interactive evolutionary algorithm for multiobjective optimization," *Helsinki School of Economics*, *Working Paper*, W-412, January 2007.

[19] T. Wagner, N. Beume, and B. Naujoks, "Pareto-, aggregation-, and indicator-based methods in many-objective optimization," *Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007*, pp. 742-756, Springer, Berlin, March 2007.

[20] E. Zitzler, D. Brockhoff, and L. Thiele, "The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration," *Lecture Notes in Computer Science 4403: Evolutionary Multi-Criterion Optimization - EMO 2007*, pp. 862-876, Springer, Berlin, March 2007.

[21] D. Brockhoff and E. Zitzler, "Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods," *Proc. of 2007 IEEE Congress on Evolutionary Computation*, pp. 2086-2093, Singapore, September 25-28, 2007.

[22] H. Ishibuchi, N. Tsukamoto, and Y. Nojima, "Iterative approach to indicator-based multiobjective optimization," *Proc. of 2007 IEEE Congress on Evolutionary Computation*, pp. 3697-3704, Singapore, September 25-28, 2007.

[23] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: multiobjective selection based on dominated hypervolume," *European Journal of Operational Research* 180 (3), pp. 1653-1669, September 2007.

[24] E. Zitzler, L. Thiele, and J. Bader, "SPAM: Set preference algorithm for multiobjective optimization," *Lecture Notes in Computer Science 5199: Parallel Problem Solving from Nature - PPSN X*, pp. 847-858, Springer, Berlin, 2008.

[25] E. Zitzler and L. Thiele, "Multiobjective optimization using evolutionary algorithms – A comparative case study," *Lecture Notes in Computer Science*, vol. 1498, pp. 292-301, Springer, Berlin, September 1998.

[26] J. D. Knowles and D. W. Corne, "On metrics for comparing non-dominated sets," *Proc. of 2002 IEEE Congress on Evolutionary Computation*, pp. 711-716, Honolulu, May 12-17, 2002.

[27] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. on Evolutionary Computation* (7) 2, pp. 117-132, April 2003.

[28] J. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," *TIK Report* 241, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, July 2005.

[29] H. Ishibuchi, T. Doi, and Y. Nojima, "Incorporation of scalarizing fitness functions into evolutionary multiobjective optimization algorithms," *Lecture Notes in Computer Science 4193: Parallel Problem Solving from Nature - PPSN IX*, pp. 493-502, Springer, Berlin, September 2006.

[30] E. J. Hughes, "MSOPS-II: A general-purpose many-objective optimiser," *Proc. of 2007 IEEE Congress on Evolutionary Computation*, pp. 3944-3951, Singapore, September 25-28, 2007.

[31] M. J. Alves and M. Almeida, "MOTGA: A multiobjective Tchebycheff based genetic algorithm for the multidimensional knapsack problem," *Computers & Operations Research* 34 (11), pp. 3458-3470, November 2007.

[32] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. on Evolutionary Computation* 11 (6), pp. 712-731, December 2007.

[33] 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* 28 (3), pp. 392-403, August 1998.

[34] A. Jaszkiewicz, "Genetic local search for multi-objective combinatorial optimization," *European Journal of Operational Research* 137 (1), pp. 50-71, February 2002.

[35] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Trans. on Evolutionary Computation* 7 (2), pp. 204-223, April 2003.

[36] T. Murata, H. Ishibuchi, and M. Gen, "Specification of genetic search directions in cellular multi-objective genetic algorithm," *Lecture Notes in Computer Science 1993: Evolutionary Multi-Criterion Optimization - EMO 2001*, pp. 82-95, Springer, Berlin, March 2001.

[37] H. Ishibuchi and S. Kaige, "Effects of repair procedures on the performance of EMO algorithms for multiobjective 0/1 knapsack problems," *Proc. of 2003 Congress on Evolutionary Computation*, pp. 2254-2261, Canberra, Australia, December 8-12, 2003.