Evaluation of Simultaneously Optimizing Multiple Models in Vehicle Design using Distributed NSGA-II Sharing Extreme Non-Dominated Solutions

Mikiko Sato, Minami Miyakawa, Hiroyuki Sato, and Yuji Sato

Abstract**—** A recent trend in multiobjective evolutionary algorithms is to increase the population size to approximate the Pareto front with high accuracy. On the other hand, the NSGA-II algorithm widely used in multiobjective optimization performs non-dominated sorting in solution ranking, which means an increase in computational complexity proportional to the square of the population. This execution time becomes a problem in engineering applications. It is also difficult to achieve high speeds while maintaining the accuracy of solution searching by simply applying fast, parallel processing to standard genetic operations. In this paper, we propose NSGA-II distributed processing in a many-core environment and a migration method that shares extreme Pareto solutions of the current generation among all cores after performing compensation of the non-dominated solution set obtained by distributed processing. Using typical real-valued functions, constrained knapsack problems, and a simultaneous optimization problem that arises in the design of multiple vehicle models for evaluation, we show that the proposed migration method can significantly reduce execution time while obtaining higher accuracy in solution searching compared with NSGA-II executed on a single CPU.

Keywords**—** multiobjective evolutionary algorithms, NSGA-II, parallel processing, many-core CPU, knapsack problem.

I. Introduction

In recent years, the trend in multiobjective evolutionary algorithms has been to increase the population size to approximate the Pareto-optimal front [1] with high accuracy [2]. Increasing the population size, however, results in an exponential increase in the computational complexity required for evaluating the dominant-subordinate relationships among solutions. As a result, execution time can be a problem when applying such an approach to engineering applications. For example, NSGA-II [1], an algorithm widely used in multiobjective optimization, performs non-dominated sorting in solution ranking resulting in a computational complexity at the very least of $O(MN^2)$ (M: number of objectives, N: population size) in every generation [3].

On the other hand, research on methods of implementing evolutionary algorithms on massively parallel computers as one means of speeding up calculations has been quite active since the 1990s [4], [5]. Research on parallel computation of evolutionary algorithms using a many-core environment as in a multicore processor or graphics processing unit (GPU) has also begun [6], [7]. These prior studies have centered on research that aims to speed up standard genetic operations through parallel processing, and they have produced effective results with respect to a variety of benchmark problems and actual applications. Here, a number of evolutionary algorithms have been proposed for multiobjective optimization, but such algorithms typified by NSGA-II, SPEA2 [8], MOEA/D [9], and NSGA-III [10] add original processing different from ordinary genetic operations to improve convergence performance and the diversity of the non-dominated solution set. The need for such processing reflects the difficulty of achieving high speeds while maintaining the accuracy of solution searching by simply applying fast, parallel processing to standard genetic operations. For example, the standard island model for parallel processing of evolutionary algorithms in a many-core environment repeats the process of dividing the population into subgroups (islands), applying standard genetic operations in parallel, and performing the migration of elite individuals within each island to other islands at appropriate times. However, when dividing non-dominated sorting among multiple islands and evaluating the elite individuals (non-dominated solutions) on each island across the entire population, they may not be truly non-dominated after all. This problem becomes especially noticeable as the number of islands is increased to speed up processing.

In response to this problem, we previously proposed a method that improves performance while maintaining the accuracy of the Pareto-optimal solution set by repeating NSGA-II distributed processing in a many-core environment as inspired by the divide-and-conquer method combined with migration processing for compensation of the non-dominated solution set obtained by distributed processing [11], [12]. We

Mikiko Sato is the specially appointed associate professor of Tokai University, Tokyo, Japan, (corresponding author to provide phone: 03-3441-1171 (1419); e-mail: mikiko.sato@ tokai.ac.jp).

Minami Miyakawa is the specially appointed researcher of Hosei University, Tokyo, Japan and the research fellow of Japan Society for the Promotion of Science.

Hiroyuki Sato is the associate professor of the University of Electro-Communications, Tokyo, Japan.

Yuji Sato is the professor of Hosei University, Tokyo, Japan.

evaluated this method with benchmark problems using a number of typical real-valued functions and showed that execution time for obtaining a Pareto-optimal solution set of equivalent hypervolume (HV) could be greatly shortened and that a higher accuracy in solution searching could be obtained. In this paper, we propose a migration method that shares extreme Pareto solutions of the current generation among all cores after performing compensation of the non-dominated solution set obtained by distributed processing. We also, using constrained knapsack problem and a simultaneous optimization problem that arises in the design of multiple vehicle models, demonstrate its effectiveness in improving diversity in solution searching.

The remainder of this paper is organized as follows. Section 2 presents the background of our research and our latest migration method of DNSGA-II, Section 3 introduces the benchmark problem of simultaneous optimization of multiple models in vehicle design, and evaluates the method using the benchmark problem. Section 4 then discusses experimental results and Section 5 concludes the paper.

II. Distributed NSGA-II

A. *Background*

Several studies on fast, parallel processing of multi-objective evolutionary algorithms have already been reported [13], [14], [19], but these have focused on research for applying standard genetic operations for single-objective problems to multi-objective problems and on research related to methods of dividing an objective space. Here, one method for dividing objective space [19] can be an extremely effective parallel high-speed method provided that the shape of the Pareto front is understood to some degree beforehand. However, if the distribution of the Pareto front should have some bias or be irregular, and if there is no prior knowledge of the shape of that Pareto front, it will then be difficult to divide the variable space or objective space appropriately. In this paper, with the aim of proposing a general-purpose distributed method that can be applied even without prior knowledge of the Pareto front distribution, we propose a distributed processing method that assumes random division of the population. On the other hand, high-accuracy multi-objective evolutionary algorithms such as NSGA-II, SPEA2, MOEA/D, and NSGA-III add original processing different from ordinary genetic operations to improve convergence and the diversity of the non-dominated solution set, which reflects the fact that simply applying the technologies of prior research cannot maintain the accuracy of solution searching. To give an example, the island model repeats the process of dividing the population into subgroups (islands), executing NSGA-II in parallel, and migrating elite individuals on each island to other islands at appropriate times. However, when executing non-dominated sorting — a feature of NSGA-II — on multiple islands in a divided manner and evaluating the elite individuals (non-dominated solutions) on each island across the entire population, the problem arises that some of those solutions may not be non-dominated after all. In this way, when generating next-generation individuals (search points) as new individuals based on solution candidates erroneously classified as non-dominated solutions in genetic operations for each subgroup, search accuracy cannot be expected to sufficiently improve compared with generating next-generation individuals based on correct non-dominated solutions. In addition, increasing the number of islands to accelerate processing increases the frequency of appearance of solution candidates erroneously classified as non-dominated solutions.

Actually, in our previous work, we showed that search accuracy (HV) tends to drop when increasing parallelism without appropriate migration for a fixed total population [12]. To address the above issue, we have proposed the method for achieving fast, parallel processing of DNSGA-II while maintaining the accuracy of the Pareto-optimal front as an efficient migration method [12], [20], [21].

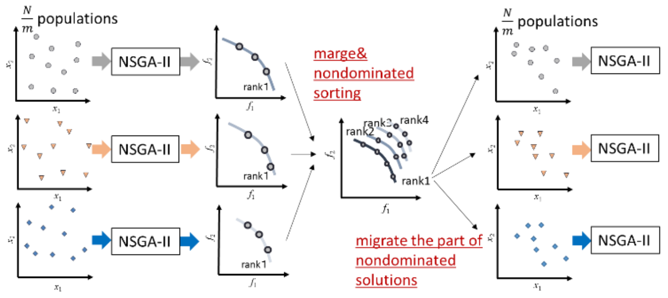

$\frac{N}{m}$ populations
NSGA-II
rank1
marge& nondominated sorting
rank2
rank3
rank4
migrate the part of nondominated solutions
x_1
x_2
f_1
f_2

Fig.1 Concept of DNSGA-II with migration.

B. *Basic Concept of DNSGA-II*

The concept of DNSGA-II with migration included is shown in Fig. 1. In the figure, the method executes NSGA-II in each subgroup in parallel, gathers the non-dominated solution sets (rank 1 solution sets) obtained by solution searching on each CPU, and again performs non-dominated sorting with ranking as compensation processing. Next, the method performs migration by allocating to each CPU a portion of the non-dominated solution set obtained by compensation. It then proceeds to the next generation of solution searching by NSGA-II on each CPU. This repeated execution of NSGA-II distributed processing in a many-core environment while performing compensation processing of the false non-dominated solution sets obtained in each subgroup achieves high-speed NSGA-II while maintaining the accuracy of solution searching. In contrast to the conventional island model, which aims to improve the accuracy of solution searching by having elite individuals in each subgroup undergo migration to other islands, the key feature of this proposal is distributed processing of NSGA-II in a many-core environment while performing compensation processing of the false non-dominated solution sets obtained in each subgroup.

C. *Migration Method for e-DNSGA-II*

To resolve the issue, we have already proposed the method "DNSGA-II [12]" for achieving fast, parallel processing of NSGAII while maintaining the accuracy of the Pareto-optimal front. Here, we propose the newly migration method sharing extreme non-dominated solutions is shown in Fig. 2. In the following, we denote DNSGA-II using this migration method as Distributed NSGA-II sharing extreme non-dominated solutions (e-DNSGA-II). In the figure, *Pi*, *Si* (*i*=1, …, *n*) indicates the proposed migration method sharing extreme non-dominated solutions first gathers the non-dominated solution sets (rank 1 solution sets) obtained by solution searching on each CPU and again performs non-dominated sorting and compensation processing with ranking. It then preferentially allocates the non-dominated solutions at both ends of the current Pareto front to all CPUs in place of the excluded false non-dominated solutions. In the event that all non-dominated solutions are true solutions after again applying non-dominated sorting, the method deletes those individuals with small crowding distance (CD) values replacing them with these non-dominated solutions at both ends of the Pareto front. In short, this method performs migration by deleting false non-dominated solutions and non-dominated solutions with small CD values and replacing them with extreme non-dominated solutions at both ends of the current Pareto front (shares extreme non-dominated solutions among all islands).

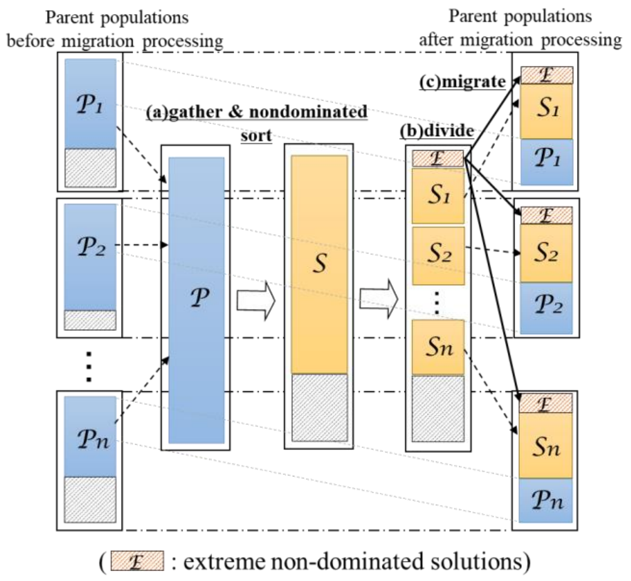

Parent populations before migration processing
Parent populations after migration processing
(a)gather & nondominated sort
(b)divide
(c)migrate
$\mathcal{P}_1$
$\mathcal{P}_2$
$\mathcal{P}_n$
$\mathcal{P}$
S
$\mathcal{E}$
S_1
S_2
S_n
($\mathcal{E}$: extreme non-dominated solutions)

Fig. 2 Compensation of non-dominated solution set and migration method sharing extreme non-dominated solutions among all islands.

D. Estimates of performance

1) No parallel, high-speed processing of nondominated sorting: Most of the execution time of NSGA-II is taken up by non-dominated sorting, and given that the computational complexity of non-dominated sorting is on the order of NSGA-II $O(MN^2)$ (*M*: number of objectives, *N*: population size) at the least [3], total computational complexity for *G* generations of genetic operations takes on the following value:

$$O(GMN^2) \qquad (1)$$

As a result, increasing the population size to improve accuracy when applying NSGA-II to engineering applications presents a problem in terms of execution time.

2) Proposed method: When executing DNSGA-II in parallel over *n* CPUs by the proposed method, the execution time required is that corresponding to a population of *N/n* instead of *N*. In addition, as non-dominated sorting is performed for only individuals of rank 1, the execution time required for compensation processing is that corresponding to αN instead of *N* where the ratio of rank-1 individuals to all individuals is denoted by ($0 < \alpha \leq 1$). Accordingly, denoting the number of migrations as m and the data transfer time for a migration as *T*, total computational complexity takes on the following value:

$$O(GM\left(\frac{N}{n}\right)^2 + mM(\alpha N)^2 + mT) \qquad (2)$$

(*G*: number of generations, *M*: number of objectives, *N*: population size, *n*: number of CPUs, m: number of migrations, α: ratio of rank-1 individuals ($0 < \alpha \leq 1$), *T*: data transfer time).

Thus, in an ideal environment in which data transfer time can be ignored, and considering the case in which compensation processing is performed only once at the end, we can expect the improvement in speed to be at the most on the order of the square of n, the degree of parallelism, compared with execution on a single CPU.

III. Evaluation

A. Evaluation using real-valued functions

1) Experimental method: To assess the effectiveness of distributed parallelization using e-DNSGA-II, we compared two items — the accuracy of solution searching and execution time — with the total population fixed, among three types of NSGA-II execution: conventional NSGA-II on a single CPU, parallel NSGA-II without migration method, and DNSGA-II with our proposed migration method (e-DNSGA-II). Furthermore, for parallel execution of NSGA-II, we examined the accuracy of solution searching and execution time while varying the degree of parallelism and the migration interval. We performed the evaluation for 2, 4, 8 and 12 degrees of parallelism with total population fixed to 2400.

In addition to the above, we used hypervolume [15] as an indicator of the accuracy of solution searching, and for test problems, we used the multiobjective optimization problems listed in Table 1 taken from the problems attached to the NSGA-II source code [16]. The test execution environment is summarized in Table 2. Experimental results were taken to be the average of 20 trials.

Table 1: Problem Instance

Problem	M	G	Generations	reference point
BNH	2	150	2	(200, 55)
ZDT1	2	200	0	(1.2, 1.5)

Table 2: Execution Environment

CPU	Intel Xeon X5680x2 (12cores, 24threads, 3.33GHz)
Memory	DDR3 24GB
OS	Linux 2.6.32
Compiler	gcc version 4.4.7

2) Experimental results: Figure 3 shows experimental results for different degrees of parallelism while keeping total population fixed. Given a total population size of 2400, we compare the accuracy of solution searching (hypervolume) and the execution time of that solution searching for conventional NSGA-II using a single CPU (in the figure, "Single core"), conventional island model (hypervolume and execution time are indicated by "hv island Model (5Mig)" and "Time island Model (5Mig)" for *N* core), parallel NSGA-II without migration method (hypervolume and execution time are indicated by the black bar "hv parallel NSGA-II (NoMig)" and blue plot "Time parallel NSGA-II (NoMig)" for *N* core), and the proposed method e-DNSGA-II (hypervolume and execution time are indicated by the green bar ("hv e-DNSGA-II (X Mig)") and orange plot ("Time e-DNSGA-II (X Mig)") for *N* core). As shown by the black bar in the figure, simply increasing the degree of parallelism, though improving execution performance, causes search accuracy to drop for these test problems.

These results revealed some differences according to degree of parallelism and frequency of migration, but in these test problems, performing migration with the proposed method tended to result in a higher hypervolume compared to parallel NSGA-II without migration method. Furthermore, the proposed method was able to keep execution time below that of single-CPU NSGA-II (in the figure, "Single core") despite the overhead associated with migration processing.

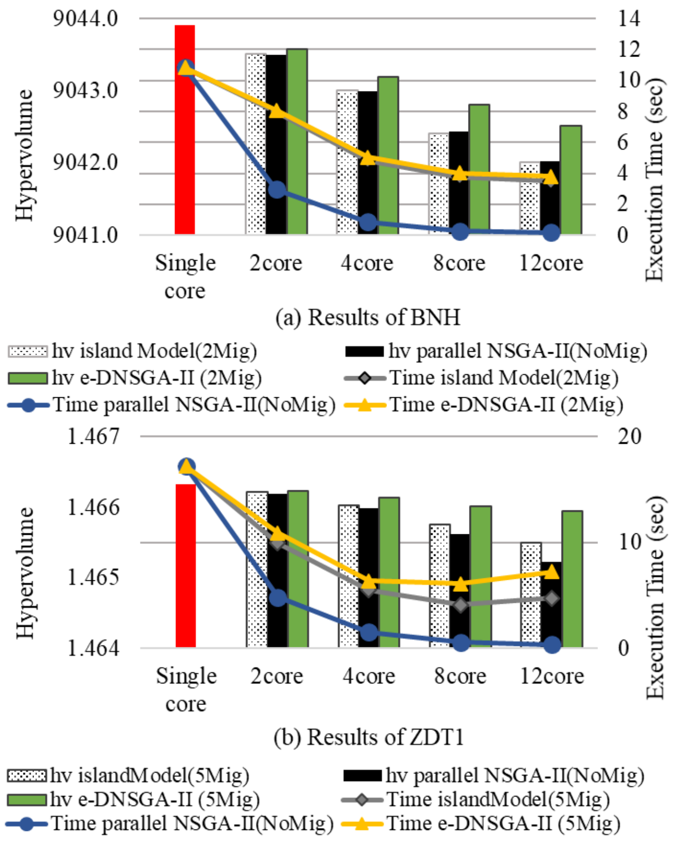


Fig. 3 Experimental results of parallel execution with NSGA-II (No migration with DNSGA-II).

B. *Evaluation using discrete optimization problems*

1) Experimental method: Using the constrained knapsack problem described below, we performed an evaluation with respect to discrete optimization problems. Considering that using this problem for an evaluation takes time and that a general island model has many design variables, we here perform a comparison evaluation targeting two items — hypervolume and execution time — for the case of executing conventional NSGA-II on a single CPU and the case of parallel execution of NSGA-II using the proposed method.

As constrained multi-objective optimization problems, we focus on *mk*-KPs [17]. The *mk*-KPs are defined in equation (3).

$$\begin{cases} Maximize\ f_j(x) = \sum_{i=1}^{n} p_{ij} \times x_j \ (j = 1,2,\dots,m) \\ Subject\ to\ \sum_{i=1}^{n} w_{il} \times x_i \le c_l \ (l = 1,2,\dots,k) \end{cases} \quad (3)$$

The problem has n items and k knapsacks, and each item i has m profits p_{ij} ($j = 1, 2, \dots, m$) and k weights w_{il} ($l = 1, 2, \dots, k$). The task is to find a set of items $x = x_1, x_2, \cdots, x_n \in \{0, 1\}^n$ maximizing m objectives while not exceeding k knapsack capacities c_l. The knapsack capacity c_l is defined in equation (4). φ is the feasibility ratio for each knapsack (constraint), and we can control the strictness of constraints and we can control the strictness of constraints by varying φ. The *mk*-KP is different from the multi-objective knapsack problem (MOKP) [18] in that the numbers of objectives m and knapsacks k can be independently determined.

$$c_l = \varphi \sum_{i=1}^{n} w_{il} \ (l = 1,2,\dots,k) \quad (4)$$

Parameters used in the experiment are listed in Table 3 and the test execution environment is summarized in Table 4. Experimental results were taken to be the average of 10 trials.

Table3: Experimental Parameters

Population Size	2400
Degree of Parallelism	1, 2, 4, 8
Number of Generations	180000
Migration Interval	50, 300, 600, 1000
Number of Objectives	2, 3
Constraint	2
Crossover Rate	0.7
Mutation Rate	0.1
Number of Items	300
Elimination Rate	0.5

Table 4: Test Execution Environment

CPU	Intel Corei7-860 (4cores, 8threads, 2.8GHz)
Memory	DDR3 8GB
OS	Ubuntu 16.04.3 LTS
Compiler	gcc version 5.4.0

2) Experimental results:

a) Experiments comparing hypervolume and execution time: For the sake of brevity, we here examine the case for a migration interval of 300. The same behavior was observed for the other migration intervals. Execution results for conventional NSGA-II using a single CPU (in the figure, "Single core"), parallel NSGA-II without migration method ("parallel NSGA-II (NoMig)"), and e-DNSGA-II ("e-DNSGA-II (300Mig)") are shown in Figs. 4 and 5 for two objectives and three objectives, respectively. The left and right axes in the figures show hypervolume and execution time, respectively. For the case of two objectives in Fig. 4, the e-DNSGA-II method proposed here achieve a high hypervolume. Similarly, for the case of three objectives in Fig. 5, the results for e-DNSGA-II show high hypervolume. Additionally, for no migration, results for both two and three objectives reveal that search accuracy (hypervolume) drops with increase in parallelism. We consider the reason for this to be that when executing non-dominated sorting on multiple islands in a divided manner the problem arises that some of the solutions may not be non-dominated after all.

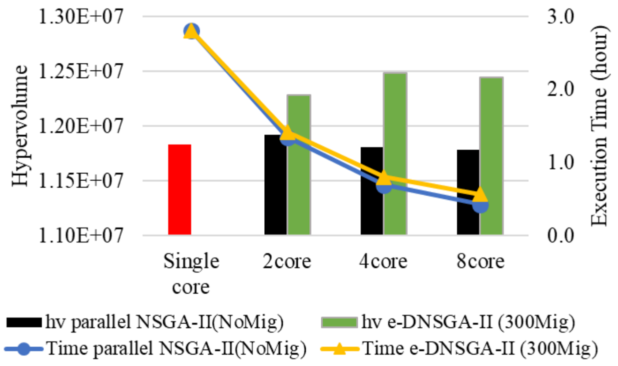


Fig. 4 Hypervolume and execution time of each method (2 objectives)

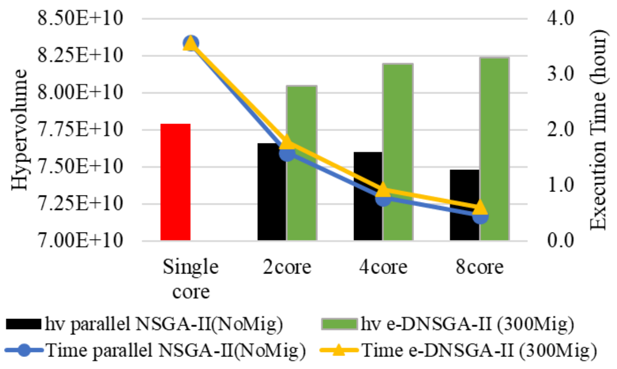


Fig. 5 Hypervolume and execution time of each method (3 objectives).

Next, for the case of executing NSGA-II on a single CPU, an optimal hypervolume could not be obtained even after eight hours. As a result, the execution time of NSGA-II is approximately 5.0 times and 5.8 times that of e-DNSGA-II (8 cores) for achieving a suboptimal solution with a hypervolume of 1.18E+0.7 and 7.8E+10 for two and three objectives, respectively. These results reflect the high-speed operation of the proposed e-DNSGA-II method.

b) Comparison of diversity and convergence by shape of Pareto-optimal front: The shapes of the Pareto-optimal front for e-DNSGA-II and single-CPU NSGA-II are compared in Figs. 6 and 7 for two and three objectives, respectively. Figure 6 plots Pareto fronts as minimization problems for the sake of calculating HV values. For the case of two objectives in Fig. 6, it can be seen that the proposed e-DNSGA-II method achieves greatly improved diversity at both extremes of the Pareto-optimal front. It can also be observed from these results that the final number of non-dominated solutions increases from the 24 of NSGA-II to the 39 of e-NSGA-II and that the latter features a uniform distribution of non-dominated solutions. Next, for the case of three objectives in Fig. 7, it can be seen that non-dominated solutions show a strong tendency to concentrate at the center of the Pareto-optimal front for execution on a single CPU. In contrast, it can be seen that search ability for non-dominated solutions at the extremes of the Pareto-optimal front improves for execution by e-DNSGA-II with the result that many non-dominated solutions are distributed over a broad range. The final number of non-dominated solutions increases from the 280 of NSGA- II to the 364 of e-NSGA-II.

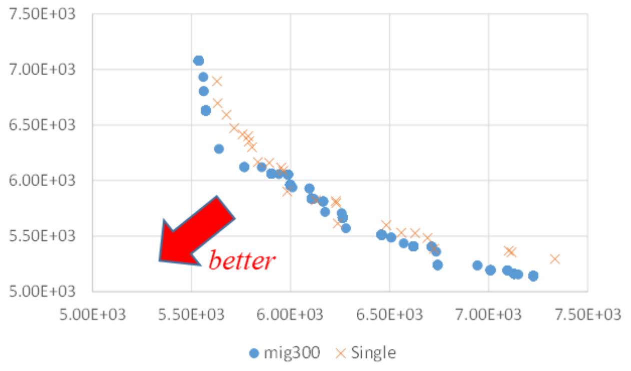


Fig. 6 Comparison of NSGA-II and e-NSGA-II Pareto fronts for 2 objectives.

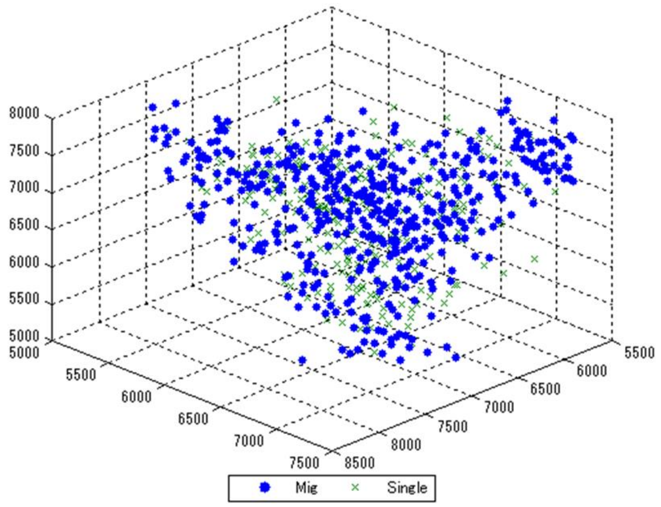


Fig. 7 Comparison of NSGA-II and e-NSGA-II Pareto fronts for 3 objectives.

C. *Evaluation using real-world problems*

1) Benchmark Test of Simultaneously Optimizing Multiple Models in Vehicle Design: In this section, we describe a simultaneous optimization problem targeting multiple models of vehicles. This benchmark test is a vehicle design problem in the context of real-world problems [22], [23]. The idea here is to optimize a design variable vector that determines the design of multiple models at once as a multiobjective problem. Compared with existing benchmark problems, the benchmark test introduced here features many vector elements and constraint functions plus inter-variable dependency with respect to objectives and constraint functions. As a result, an increase in computation time for calculating optimal solutions presents a problem.

The design variable vector is defined as $x = (x_1, x_2, \cdots, x_K)$ whose elements are real values x_i (i = 1, …, K). Here, K is the product $\alpha = C \times d$ where C is the number of vehicle models to be simultaneously optimized and d is the number of design variables per vehicle. The configuration of the design variable vector is shown in Fig. 8. The design variables for each car are arranged in the order of common structural components. That is, the design variable that determines the p_{th} common structural component coincides with x_p of the first car, x_{p+d} of the second car, and x_{p+2d} of the third car. In this way, a single design variable vector x simultaneously determines the design of C vehicle models. The objective functions to be optimized are as follows:

Minimize

$$f_1(x) = \sum_{l=1}^{C} Mass(x_{1+(l-1)d}, x_{2+(l-1)d}, \cdots, x_{d+8l-1)d}) \qquad (5)$$

Maximize

$$f_2(x) = \left|\{p \in \{1, \cdots, d\} x_p \cong x_{p+d} \cong x_{p+2d} \cong \cdots \cong x_{p+(C-1)d}\}\right| \qquad (6)$$

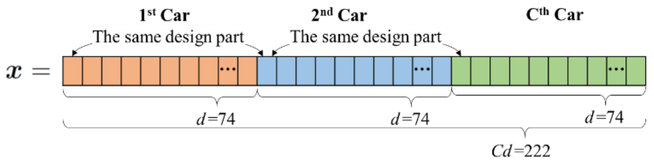


Fig.8 Design variables.

The first objective is to minimize the total mass of Mass function calculates the mass of each car based on a multiple regression model. The second objective is to maximize the number of common design parts among C cars. Here, design variables that determine individual structural components are judged to be common if they are the same among C cars. Specifically, design variables with a difference of less than 0.05 in their values are considered to be the same ($\simeq$). For this benchmark test, we set C = 3 and d = 74 so that the number of elements making up design variable vector is $K = C \times d$ = 222. In addition, the total number of constraint functions calculated by the response surface method and constraint functions related to magnitude relationships among variables as dictated by design requirements is k = 54. In this paper, we use the following equation to handle the second objective as a minimization problem for convenience sake.

Minimize

$$f'_2(x) = d - f_2(x) \qquad (7)$$

2) Experimental method: To assess the effectiveness of distributed parallelization using e-DNSGA-II, we compared two items — the accuracy of solution searching and execution time — with the total population fixed, among two types of NSGA-II execution: conventional NSGA-II on a single CPU, and e-DNSGA-II with our proposed migration method. Furthermore, for parallel execution of NSGA-II, we examined the accuracy of solution searching and execution time while varying the degree of parallelism and the migration interval. We performed the evaluation for 6 and 12 degrees of parallelism with total population fixed to 1200. The test execution environment is summarized in Table 5. Table 6 shows the setting parameters of NSGA-II.

In addition to the above, we used hypervolume [15] as an indicator of the accuracy of solution searching. HV is the area in objective space determined from the obtained solution set and a reference point. A higher HV for an obtained solution set means that the search could be performed with greater accuracy. In this paper, we used the same normalization method for objective function values as described in [22] and used [1.1, 0.0] as the reference point. Experimental results were taken to be the average of 10 trials.

Table 5: Test Execution Environment

CPU	Intel(R) Core(TM) i9-7920X CPU (12cores, 24threads, 2.90GHz)
Memory	32GB DDR4 SDRAM
OS	Ubuntu 16.04.4 LTS
Compiler	gcc version 5.4.0

Table 6: Parameter of NSGA-II

Parameter	Volume
Population Size	100, 200, 400 1200(100pop*12core, 200pop*6core, 400pop*3core)
Number of Generation	600
Crossover Probability	1.0
Crossover Distribution Index	10
Mutation Probability	0.05
Mutation Distribution Index	10

3) Experimental results: Experimental results are shown in Figs. 9 and 10. For a fixed total population of 1200 divided into 12 sets of 100 individuals each, Fig. 9 compares HV values for the same time but different migration intervals. In the figure, we denote conventional NSGA-II executed on a single CPU for a total population of 1200 as "single," and the proposed e-DNSGA-II with migration performed every *X* generations for a total population of 1200, respectively. Fig. 10 presents HV versus generations for e-DNSGA-II with 1200 individuals divided up equally among 12 CPUs (100 individuals per CPU) and for conventional NSGA-II using a single CPU with a total population of 1200. We show for reference purposes the results obtained by single-CPU NSGA-II executed up to 50 generations for 100 individuals. Similarly, Figs. 11 and 12 show experimental results for a fixed total population of 1200 divided into 6 sets of 200 individuals each.

For a fixed total population of 1200, these results show that distributed processing by e-DNSGA-II would tend to have a higher HV value than single-CPU NSGA-II for the same time. Furthermore, despite the processing overhead for migration, they also show that e-DNSGA-II obtained higher HV values than single-CPU NSGA-II for nearly the same execution time regardless of the migration interval. In any case, it can be seen that the proposed method is 5 to 10 times faster than conventional single-CPU NSGA-II with respect to the time taken to achieve an HV value of 0.05. Furthermore, it can be seen from Figs. 10 and 12 that no major difference in HV value per generation could be observed between single-CPU generation and parallel execution for both 6 and 12 CPUs.

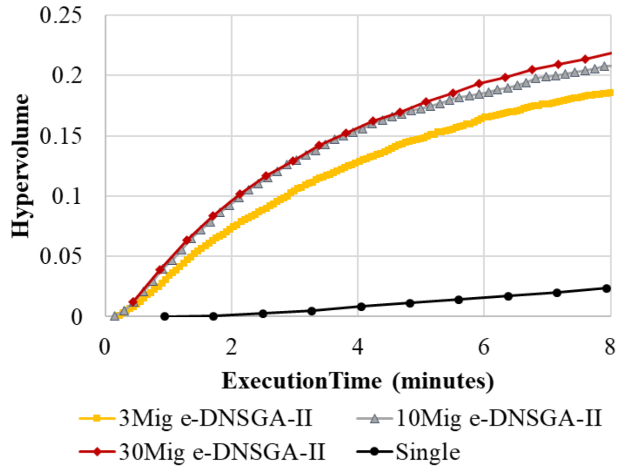


Fig.9 Hypervolume transition by execution time (100 populations per core).

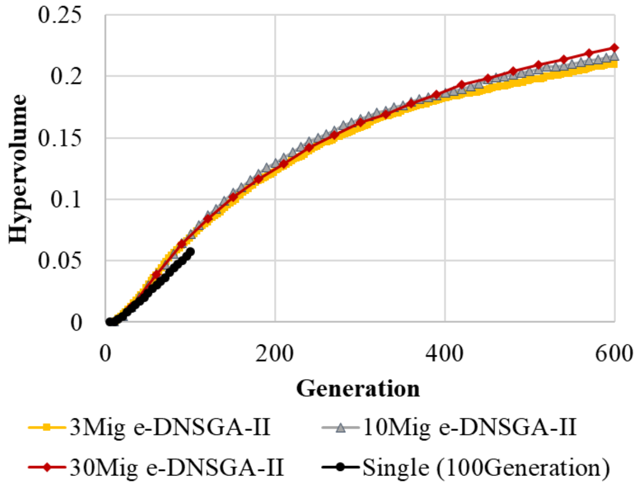


Fig. 10 Hypervolume transition by generation (100 populations per core).

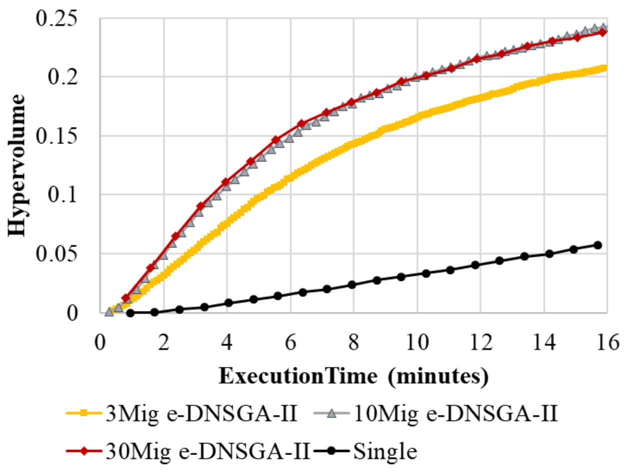


Fig.11 Hypervolume transition by execution time (200 populations per core).

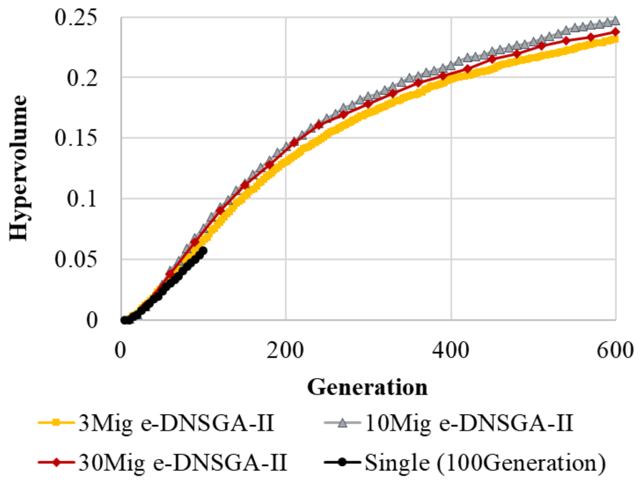
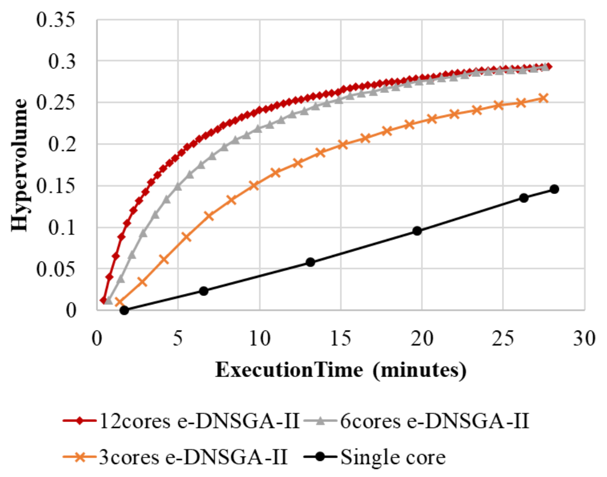


Fig. 12 Hypervolume transition by generation (200 populations per core).

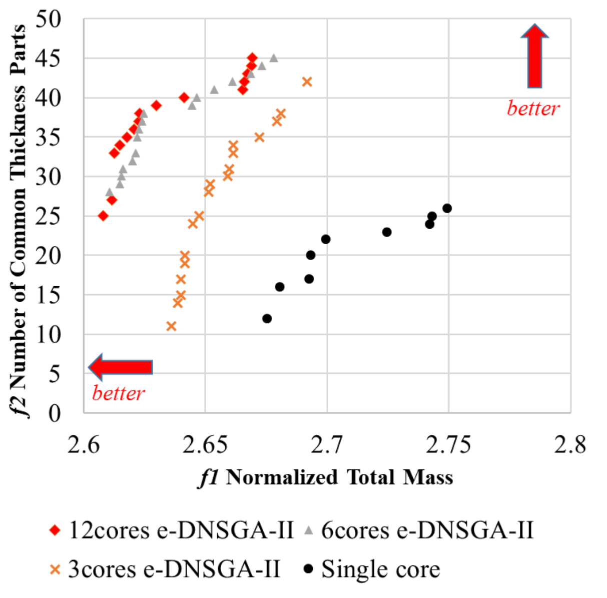
The HV values for the same time and the shapes of the Pareto fronts for the case of 12 CPUs (100 individuals per CPU), the case of 6 CPUs (200 individuals per CPU), and the case of 3 CPUs (400 individuals per CPU) are shown in Fig. 13. These results compare the convergence of Pareto fronts for different degrees of parallelism given a migration interval of 30 generations on extending execution time to about 30 minutes. It can therefore be seen from Fig. 13 that Pareto front convergence improves as the degree of parallelism increases when compared over the same execution time.

IV. Discussion

First, for the BNH and ZDT1 test problems, we compared results for the same number of generations and found that conventional NSGA-II using a single CPU was apt to exhibit higher performance in solution searching (higher hypervolume). However, the use of migration whatever the degree of parallelism was apt to increase the accuracy of solution searching approaching that of single-CPU NSGA-II. We consider that these test problems tend to need a sufficiently large population per CPU, and if dividing into small populations, that incorporating solutions from other CPUs by migration is effective. However, when increasing the degree of parallelism



(a) Hypervolume transition by execution time



(b) Pareto fronts after 30 min.

Fig.13 Comparison of obtained solution sets for e-DNSGA-II with migration every 30 generations and single-CPU NSGA-II.

while keeping the population per CPU fixed, the same hypervolume values as that of single-CPU NSGA-II were obtained by the proposed method. On comparing execution-time performance for the same hypervolume values here, an improvement greater than 10 times was achieved.

Second, in the evaluation using the constrained knapsack problem, it was seen for the case of no migration that search accuracy tended to drop by distributed processing. We explain the reason for this in conjunction with the evaluation results using real-valued functions. For problems that include singularities and problems that require a particularly high level of diversity, dividing the population among multiple cores and performing distributed processing will improve diversity and raise the hypervolume value that is eventually obtained. However, we found in this evaluation that an improvement in diversity generally came at the expense of a drop in convergence ability and that the hypervolume value that was eventually obtained tended to drop. On the other hand, we consider that applying appropriate migration as in the case of e-DNSGA-II has the effect of compensating for this drop in convergence ability and that, in addition to improving diversity, has the capability of simultaneously achieving high-speed parallel processing and improving search ability (improving hypervolume) in the end.

Next, the results of Figs. 9 and 11 show that the proposed method obtains the same HV at extremely high speed compared with execution by conventional single-CPU NSGA-II. Furthermore, a comparison of Figs. 13 suggests that an increase in parallelism can be associated with a high speed-up ratio. Although some difference can be seen in search performance according to the migration interval, there appears to be no major difference in the tendency to speed up.

The difficulty of determining a Pareto front with good accuracy in a real-world problem within a realistic period of time is not limited to the simultaneous optimization problem taken up here in the design of multiple vehicle models. We therefore consider the proposed method to be effective in dealing with this issue since it can improve accuracy in solution searching while simultaneously reducing execution time. In addition, the proposed method can be used to achieve fast, parallel processing of not just NSGA-II but any algorithm using non-dominated sorting. In future research, we plan to investigate scalability for even higher degrees of parallelism and conduct detailed comparison experiments with the standard island model.

V. Conclusion

In this paper, we proposed e-DNSGA-II, a distributed parallel processing method that repeatedly performs NSGA-II distributed processing in a many-core environment, compensation of the non-dominated solution set obtained by distributed processing, and migration by sharing among all cores the solutions at the extreme points of the current generation's Pareto front. Using two typical real-valued-function problems, a constrained knapsack problem, and a simultaneous optimization problem occurring in the design of multiple vehicle models, we compared the proposed method with single-CPU NSGA-II and parallel NSGA-II without migration method in terms of the accuracy of solution searching and execution time. We showed that the proposed method has the effect of improving diversity and the ability of improving performance considerably while maintaining the accuracy of solution searching.

Acknowledgment

The authors would like to extent our sincere gratitude to Takeshi Kohira for providing the vehicle benchmark which is an optimization benchmark with the actual engineering design features of the car-body structural development.

REFERENCES

[1] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” IEEE Transactions on Evolutionary Computation, vol. 6, no. 2, pp. 182–197, Apr 2002.

[2] H. Ishibuchi, N. Akedo, and Y. Nojima, “Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems,” IEEE Transactions on Evolutionary Computation, vol. 19, no. 2, pp. 264–283, April 2015.

[3] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, “An efficient approach to nondominated sorting for evolutionary multiobjective optimization,” IEEE Transactions on Evolutionary Computation, vol. 19, no. 2, pp. 201–213, April 2015.

[4] V. S. Gordon and L. D. Whitley, “Serial and parallel genetic algorithms as function optimizers,” in Proceedings of the 5th International Conference on Genetic Algorithms. Morgan Kaufmann Publishers Inc., 1993, pp. 177–183.

[5] H. Mühlenbein, “Parallel genetic algorithms population genetics and combinatorial optimization,” in Proceedings of the 3rd International Conference on Genetic Algorithms. Morgan Kaufmann Publishers Inc., 1989, pp. 416–421.

[6] J. H. Byun, K. Datta, A. Ravindran, A. Mukherjee, and B. Joshi, “Performance analysis of coarse-grained parallel genetic algorithms on the multi-core sun UltraSPARK T1,” in IEEE Southeastcon 2009, March 2009, pp. 301–306.

[7] R. Serrano, J. Tapia, O. Montiel, R. Sepulveda, and P. Melin, “High Performance Parallel Programming of a GA Using Multi-core Technology,” Springer Berlin Heidelberg, 2008, pp. 307–314.

[8] E. Zitzler, M. Laumanns, and L. Thiele, “Spea2: Improving the strength pareto evolutionary algorithm,” Tech. Rep., 2001.

[9] Q. Zhang and H. Li, “Moea/d: A multiobjective evolutionary algorithm based on decomposition,” IEEE Transactions on Evolutionary Computation, vol. 11, no. 6, pp. 712–731, Dec 2007.

[10] K. Deb and H. Jain, “An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints,” IEEE Transactions on Evolutionary Computation, vol. 18, no. 4, pp. 577–601, Aug 2014.

[11] Y. Sato, M. Sato, and M. Miyakawa, “Distributed NSGA-II with migration using compensation on many-core processors for improving performance and accuracy,” in Proceedings of the Genetic and Evolutionary Computation Conference Companion, ser. GECCO ’17. New York, NY, USA: ACM, 2017, pp. 161–162. [Online]. Available: http://doi.acm.org/10.1145/3067695.3075974

[12] Y. Sato, M. Sato and M. Miyakawa, "Distributed NSGA-II using the divide-and-conquer method and migration for compensation on manycore processors," 2017 21st Asia Pacific Symposium on Intelligent and Evolutionary Systems (IES), Hanoi, 2017, pp. 83-88.

[13] T. Hiroyasu, M. Miki, and S. Watanabe, “Divided range genetic algorithms in multiobjective optimization problems,” in Proceedings of International Workshop on Emergent Synthesis (IWES99, 1999, pp. 57–66.

[14] F. de Toro, J. Ortega, J. Fernandez, and A. Diaz, “Psfga: a parallel genetic algorithm for multiobjective optimization,” in Proceedings 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing, 2002, pp. 384–391.

[15] N. Beume, C. M. Fonseca, M. Lopez-Ibanez, L. Paquete, and J. Vahrenhold, “On the complexity of computing the hypervolume indicator,” IEEE Transactions on Evolutionary Computation, vol. 13, no. 5, pp. 1075–1082, Oct 2009.

[16] S. D. at KanGAL, Multi-objective NSGA-II code in C, Revision 1.1.6, July 2011, http://www.iitk.ac.in/kangal/codes.shtml.

[17] D. P. Hans Kellerer, Ulrich Pferschy, Knapsack Problems. Springer, 2004

[18] E. Zitzler, Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Technology, Zurich, 1999.

[19] H. Liu, F. Gu and Q. Zhang, "Decomposition of a Multiobjective Optimization Problem Into a Number of Simple Multiobjective Subproblems," in IEEE Transactions on Evolutionary Computation, vol. 18, no. 3, pp. 450-455, 2014.

[20] Y. Sato, M. Sato, and M. Miyakawa, “Distributed NSGA-II Sharing Extreme Non-dominated Solutions,” in Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO ’18, Late-Breaking Papers. New York, NY, USA: ACM, 2018, pp. 69–70.

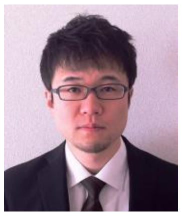
[21] Y. Sato, M. Sato, H. Goto, and M. Miyakawa, “Distributed NSGA-II Sharing Extreme Non-dominated Solutions for Constrained Knapsack Problems,” in Proceedings of the 2018 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO 2018), IEEE CPS. (to appear)..

[22] T. Kohira, H. Kemmotsu, A. Oyama, and T. Tatsukawa, “Proposal of Benchmark Problem Based on Real-World Car Structure Design Optimization,” In Proceedings of the Genetic and Evolutionary Computation Conference, GECCO ’18, pp. 183-184, 2018.

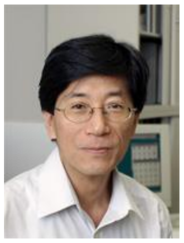
[23] M. Miyakawa, H. Sato, H. Matsumoto, M. Tanaka, M. Sato and Y. Sato, “Effects of Duplication Operator in Evolutionary Simultaneous Design Optimization of Multiple Cars,” in Proceedings of the 2018 Joint 10th International Conference on Soft Computing and Intelligent Systems and 19th International Symposium on Advanced Intelligent Systems (SCIS&ISIS 2018). IEEE Press.

Mikiko Sato received B.E., M.E., and Ph.D. degree in Engineering from Tokyo University of Agriculture and Technology, Japan in 1988, 1990 and 2006, respectively. She is Research Associate Professor of School of Information and Telecommunication Engineering, Tokai University. Her research interests include soft computing and operating system for parallel computers, many-core systems and embedded systems. She is a member of IEEE, ACM/SIGEVO, the IPSJ and the IEICE.

Minami Miyakawa received B.E., M.E., and Ph.D. degrees from The University of Electro-Communications in Tokyo, Japan in 2011, 2013 and 2016, respectively. She is a research fellow in Japan Society for the Promotion of Science (JSPS) and researching at Hosei University. Her research interests include constraint-handling in evolutionary multi-objective optimization and its applications. She received a young researcher award from IEEE Computational Intelligence Society Japan Chapter in 2013 and a best paper award from Transaction of the Japanese Society for Evolutionary Computation in 2015.



Hiroyuki Sato received B.E. and M.E. degrees from Shinshu University, Japan, in 2003 and 2005, respectively. In 2009, he received Ph. D. degree from Shinshu University. He has worked at The University of Electro-Communications since 2009. He is currently an associate professor. He received best paper awards on the EMO track in GECCO 2011 and 2014, Transaction of the Japanese Society for Evolutionary Computation in 2012 and 2015. His research interests include evolutionary multi- and many-objective optimization, and its applications. He is a member of IEEE, ACM/SIGEVO.



Yuji Sato received the BE and PhD degrees in Engineering from the University of Tokyo, Japan in 1981 and 1997, respectively. From 1981 to 2000, he was with Hitachi Ltd, Tokyo, Japan. In April 2000, he joined the Faculty of Computer and Information Sciences at the Hosei University, Japan, as an Associate Professor, and became a Professor in April 2001. He received the 2014 Highly Commended Paper Award of International Journal of Intelligent Computing and Cybernetics. His current research areas include distributed evolutionary multi-objective optimization on many-core architecture and evolution of machine learning techniques in design. He is a member of IEEE, ACM/SIGEVO, IPSJ.