Effects of Chain-Reaction Initial Solution Arrangement in Decomposition-Based MOEAs

Hiroyuki Sato[1] Minami Miyakawa[2] Keiki Takadama[1]

[1] Graduate School of Informatics and Engineering, The University of Electro-Communications, Japan
[2] Faculty of Computer and Information Sciences, Hosei University (JSPS Research Fellow), Japan

Abstract

For solving multi-objective problems, MOEA/D employs a set of weight vectors determining search directions and assigns one solution for each weight vector. Since the conventional MOEA/D assigns a randomly generated initial solution for each weight vector without considering its position in the objective space, mismatched pairs of initial solution and weight are generated, and it causes inefficient search. To enhance MOEA/D based multi-objective optimization, this work proposes a method arranging randomly generated initial solutions to weight vectors based on positions of their solutions in the objective space. The proposed method is combined with the conventional MOEA/D and MOEA/D-CRU, and their search performances are verified on continuous DLTZ4 benchmark problems with 2-5 objectives and different problem difficulty parameters. The experimental results show that the proposed method improves the search performances of MOEA/D and MOEA/D-CRU especially on problems with the difficulty to obtain uniformly distributed solutions in the objective space.

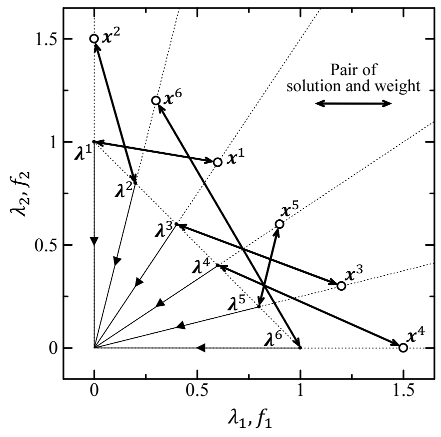
Keywords: multi-objective optimization, many-objective optimization, evolutionary algorithm, MOEA/D

1 Introduction

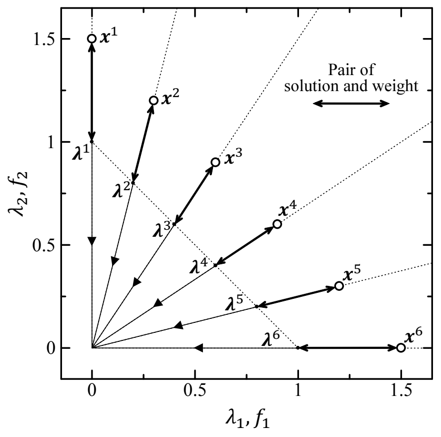
The aim of multi-objective optimization is to finely approximate the Pareto front, the optimal trade-off among conflicting objectives, with a set of solutions. The population based evolutionary algorithms is a promising approach for multi-objective optimization since a set of solutions to approximate the Pareto front can be picked from the population in a single run (Deb, 2001). Evolutionary multi-objective optimization has been intensively studied so far, and it has been known that the Pareto dominance based NSGA-II (Deb et al., 2002b), SPEA2 (Zitzler et al., 2001), the indicator based IBEA (Zitzler et al., 2004), SMS-EMOA (Beume et al., 2007), HypE (Bader et al., 2011), the decomposition based NSGA-III (Deb et al., 2014), MSOPS (Hughes, 2005), and MOEA/D (Zhang et al., 2007) are representative algorithms. Recently, the decomposition approach is being recognized as an effective approach for solving many-objective problems with more than three objectives. This work focuses on MOEA/D (Zhang et al., 2007) as one of algorithms based on the decomposition approach and tries to improve its search performance.

MOEA/D decomposes a multi-objective problem into a number of single-objective problems and simultaneously optimizes them with a single population. MOEA/D generates a set of weight vectors specifying the decomposition intervals of the objective space. Each weight vector specifies a part of the Pareto front to be approximated. To approximate each part of the Pareto front, MOEA/D assigns one solution for each weight vector. That is, each solution has the role to approximate a part of the Pareto front specified by its weight vector. Before the search, MOEA/D repeats to randomly generate an initial solution for each weight vector. Each initial solution has its own characteristic objective vector and position in the objective space, however, the conventional MOEA/D just sequentially assigns an initial solution to each weight vector without considering its position in the objective space. Consequently, several mismatched pairs of initial solution and weight are generated, and it causes inefficient search. The search performance of MOEA/D would be improved by arranging each initial solution to an appropriate weight vector close to the approximative direction of its initial solution in the objective space.

To improve the search performance of MOEA/D based algorithms, this work proposes a method arranging randomly generated initial solutions to weight vectors based on their positions in the objective space. To arrange the initial solutions, we extend the chain-reaction solution update method (Sato, 2016) previously proposed to update existing solutions in the population with generated offspring. The proposed method assigns each initial solution to a weight vector close to its approximative direction in the objective space. The proposed method makes more appropriate pairs of solution and weight than the conventional method, and the arranged solutions contribute to improving the search performance. This work uses the continuous DTLZ4 benchmark problems with 2-5 objectives and several problem difficulty parameters and verifies the effects of the proposed initial solution arrangement by combined with the conventional MOEA/D (Zhang et al., 2007) and MOEA/D-CRU (Sato, 2016).



(a) Conventional random pairs



(b) Proposed arranged pairs

Figure 1. Pairs of solution x and weight λ in the initial population

Algorithm 1 Main MOEA/D Framework (Zhang et al., 2007)

```
Input: the number of objectives $m$, the decomposition parameter $H$, the number of weight vectors and solutions in the population $N$, the neighborhood size $T$
Output: the non-dominated set of solutions
 1: $\mathscr{L} = \{\lambda^1, \dots, \lambda^N\} \leftarrow$ Generate weight vectors ($H$, $m$)
 2: for each $\lambda^i \in \mathscr{L}$ do
 3:     $\mathscr{B}_i = \{i_1, \dots, i_T\} \leftarrow$ Find nearest neighbor weight indices
 4: end for
 5: $\mathscr{P} \leftarrow$ Initialize the population        ▷ Algorithm 2 or Algorithm 4
 6: repeat
 7:     for each $i \in \{1, 2, \dots, N\}$ do
 8:         $p_1, p_2 \leftarrow$ Randomly select parent indices ($\mathscr{B}_i$)
 9:         $y \leftarrow$ Generate offspring ($x^{p_1}, x^{p_2}$)
10:         Solution Update ($y$, $i$)        ▷ Algorithm 3 or Algorithm 5
11:     end for
12: until The termination criterion is satisfied
13: return The non-dominated solutions picked from $\mathscr{P}$
```

2 MOEA/D

2.1 Algorithm

MOEA/D decomposes a multi-objective optimization problem into a number of single-objective optimization problems and simultaneously optimizes them to approximate the Pareto front of the original multi-objective problem. Algorithm 1 is a pseudocode of the MOEA/D algorithm framework. The original MOEA/D (Zhang et al., 2007) uses Algorithm 2 at 5th line for the population initialization and Algorithm 3 at 10th line for the solution update. In the following, the algorithm of the original MOEA/D is briefly described.

To decompose a m-objective problem, MOEA/D generates $N = C^{m-1}_{H+m-1}$ kinds of weight vectors $\mathscr{L} = \{\lambda^1, \lambda^2, \dots, \lambda^N\}$ based on the simplex-lattice design with the decomposition parameter H. Each weight vector λ^i specifies a part of the Pareto front to be approximated, its elements $\lambda^i_1, \lambda^i_2, \dots, \lambda^i_m$ are one of $\{0/H, 1/H, \dots, H/H\}$,

Algorithm 2 Conventional Population Initialization (Zhang et al., 2007)

```
1: procedure CONVENTIONAL POPULATION INITIALIZATION
2:     $\mathscr{P} \leftarrow \emptyset$        ▷ Population
3:     for each $i \in \{1, 2, \dots, N\}$ do
4:         $x^i \leftarrow$ Randomly generate
5:         $\mathscr{P} \leftarrow \mathscr{P} \cup x^i$
6:     end for
7:     return $\mathscr{P}$
8: end procedure
```

and unique weight vectors satisfying $\sum_{j=1}^{m} \lambda^i_j = 1.0$ are employed for the search. For each weight vector λ^i, a randomly generated solution x^i is assigned, and totally N solutions become the population $\mathscr{P} = \{x^1, x^2, \dots, x^N\}$. To select parent solutions and update solutions with newly generated offspring, MOEA/D focuses on a weight vector and its neighbor weight vectors. For each weight vector λ^i, its T-neighbors' weight indices $\mathscr{B}_i = \{i_1, i_2, \dots, i_T\}$ are stored before the search. In the search process, MOEA/D focuses on a weight vector λ^i, selects two parent solutions from solutions assigned to neighbor weights $\mathscr{B}_i$ of the focused weight λ^i, generates an offspring from the selected parents, and tries to update the neighbor solutions with the generated offspring. To compare solutions, a scalarizing function is used. This work employs the weighted Tchebycheff scalarizing function (Li et al., 2014) defined by the following equation.

$$\text{Minimize } g(x|\lambda) = \max_{1 \le j \le m} \{|f_j(x) - z_j|/\lambda_j\}, \tag{1}$$

where, z is the obtained ideal point, and its each element z_i is the minimum i-th objective value found during the search. $\lambda_j = 0$ is exceptionally replaced with $\lambda_j = 10^{-6}$ to avoid the division by zero.

2.2 Focused Issue

Each weight vector determines a search part of the Pareto front, and its solution assigned to the weight vector tries

Algorithm 3 Conventional Update (Zhang et al., 2007)

```
Input: offspring $y$, the focused index $i$
1: procedure CONVENTIONAL UPDATE ($y$, $i$)
2:     for each $j \in \mathscr{B}_i$ do
3:         if $g(y|\lambda^j)$ is better than $g(x^j|\lambda^j)$ then
4:             $x^j \leftarrow y$
5:         end if
6:     end for
7: end procedure
```

Algorithm 4 Proposed Population Initialization

```
1: procedure PROPOSED POPULATION INITIALIZATION
2:     $\mathscr{Y} \leftarrow \emptyset$                    ▷ Temporal population pool
3:     for each $i \in \{1,2,\ldots,N\}$ do
4:         $y^i \leftarrow$ Randomly generate
5:         $\mathscr{Y} \leftarrow \mathscr{Y} \cup y^i$
6:     end for
7:     $\mathscr{P} \leftarrow \emptyset$                    ▷ Population
8:     for each $i \in \{1,2,\ldots,N\}$ do
9:         CHAIN-REACTION ARRANGE AND UPDATE ($y^i$)
10:    end for
11:    return $\mathscr{P}$
12: end procedure
```

to approximate the specified part of the Pareto front. Therefore, for each weight vector, an appropriate solution should be paired are different. However, when the initial solutions are generated, the conventional MOEA/D does not consider pair matchings between solution and weight vector. According to Algorithm 2, the conventional MOEA/D repeats to assign a randomly generated initial solution for each weight vector without considering its position in the objective space. Therefore, several mismatched pairs of solution and weight are generated. Figure 1 (a) shows an example of $N = 6$ initial pairs of solution and weight generated by the conventional MOEA/D with Algorithm 2. This figure shows a two-dimensional weight space ($\lambda_1 - \lambda_2$) and an objective space ($f_1 - f_2$) which both objective functions should be minimized. A set of weight vectors $\lambda^1, \lambda^2, \ldots, \lambda^6$ and a set of initial solutions $x^1, x^2, \ldots, x^6$ are shown in this figure, and six double-headed arrows indicate pair relations of solution and weight vector. In this figure, x^6 has the second worst (highest) value of f_2 among all solutions. However, x^6 is assigned to λ^6 to find the minimum value of f_2 on the Pareto front. If x^4 having the minimum f_2 among all solutions is assigned to λ^6 instead of x^6, the search directed by λ^6 would be enhanced. Since Figure 1 (a) also includes other mismatched pairs of solution and weight, the search to find the Pareto front would be inefficient. Although the initial solutions must be generated randomly, the search would be efficient by arranging each of initial solution to its appropriate weight vector as shown in Figure 1 (b).

3 Proposed Method: Chain-Reaction Initial Solution Arrangement

3.1 Aim and Concept

To improve the search performance of MOEA/D based algorithms by enhancing a number of single objective searches directed by weight vectors, this work proposes a method appropriately arranging randomly generated initial solutions to weight vectors. Since the conventional method just sequentially assigns each initial solution to a weight vector without considering its position in the objective space, and several mismatched pairs of solution and weight are obtained as shown in Figure 1 (a). On the other hand, the proposed method tries to assign each initial solution to an appropriate weight vector with considering its approximative direction, the objective balance vector, in the objective space. Consequently, more appropriate pairs of solution and weight are obtained as shown in Figure 1 (b).

3.2 Method

To arrange the initial solutions, this work extends the chain-reaction solution update method (Sato, 2016). The chain-reaction solution update effectively replaces existing solutions in the population with generated offspring by adaptively determining target existing solutions to be updated based on the position of generated offspring. In the previous work, the chain-reaction solution update is used only for the update of existing solutions with generated offspring. To appropriately arrange initial solutions, this work extends the chain-reaction update, and utilize it in the process at 5th line of Algorithm 1 as an alternative of the conventional Algorithm 2. Algorithm 4 is the pseudocode of the chain-reaction solution arrangement proposed in this work. Algorithm 4 performs Algorithm 5 which is the extended chain-reaction solution update procedure (Sato, 2016) also for the initial solution arrangement. In Algorithm 5, 9-13th lines are newly added to the previously proposed chain-reaction solution update procedure. Since newly added 9-13th lines in Algorithm 5 do not affect to the solution update process, the same Algorithm 5 can be employed also in the solution update process.

The proposed method performs Algorithm 4 at 5th line of Algorithm 1 instead of Algorithm 2. The conventional Algorithm 2 generates the population $\mathscr{P}$ by repeating the random generation of an initial solution to be paired with each weight vector N times. Thus, the conventional method does not care about the positions of randomly generated solutions in the objective space and just sequentially assigns them to weight vectors. On the other hand, to check the relative position and approximative direction of each initial solution in the objective space, the proposed Algorithm 4 first randomly generates N solutions y^i ($i = 1, 2, \ldots, N$) and temporally stores them in the temporal population $\mathscr{Y}$ before assigning them to weight vectors

Algorithm 5 Proposed Chain-Reaction Arrange and Update

```
Input: solution $y$
 1: procedure CHAIN-REACTION ARRANGE AND UPDATE ($y$)
 2:     $b \leftarrow$Calculate balance of objective values ($y$)          ▷ Eq. (2)
 3:     $\mathscr{D} \leftarrow \{d^1, d^2, \dots, d^N\}$          ▷ Distances from all weight vectors
 4:     for each $i \in \{1, 2, \dots, N\}$ do
 5:         $d^i \leftarrow$ Calculate Euclidean distance ($b, \lambda^i$)
 6:     end for
 7:     $\mathscr{D} \leftarrow$ Sort elements in ascending order ($\mathscr{D}$)
 8:     for each $d^j \in \mathscr{D}$ do
 9:         if $x^j$ is not exist in the population $\mathscr{P}$ then
10:             $x^j \leftarrow y$
11:             $\mathscr{P} \leftarrow \mathscr{P} \cup x^j$
12:             break
13:         end if
14:         if $g(y|\lambda^j)$ is better than $g(x^j|\lambda^j)$ then
15:             $tmp \leftarrow x^j$          ▷ Preserve temporally
16:             $x^j \leftarrow y$
17:             CHAIN-REACTION ARRANGE AND UPDATE ($tmp$)
18:                                  ▷ Call recursively
19:             break
20:         end if
21:     end for
22: end procedure
```

(3-6 lines). Next, the proposed method makes pairs of solution and weight by repeating Algorithm 5 for each y^i in the temporal population $\mathscr{Y}$ (8-10th lines).

For each initial solution y in the temporal population $\mathscr{Y}$, Algorithm 5 calculates its objective balance vector b by the following equation (2nd line).

$$b_j = \frac{f_j(y) - z_j}{\sum_{\ell=1}^{m} \{f_\ell(y) - z_\ell\}} \quad (j = 1, 2, \dots, m). \tag{2}$$

Next, the proposed method calculates the Euclidean distances between the balance vector b and all weight vectors λ^i ($i = 1, 2, \dots, N$) and sorts their distances in ascending order (3-7th lines). Then, the proposed method tries to arrange y to a weight vector in short-distance order (8-21th lines). If solution x^j paired with weight vector λ^j which is the closest to the balance vector b still does not exist in the population $\mathscr{P}$, y is assigned to λ^j ($x^j \leftarrow y$). Otherwise, scalarizing function values of x^j and y are compared in the same way of the chain-reaction solution update (Sato, 2016). If y shows better scalarizing function value than the existing x^j, y becomes new x^j, and Algorithm 5 is recursively performed with the previous x^j like a chain-reaction. For each initial solution y, the above procedure is repeated in the short-distance order of the objective valance vector b and weight vectors until y is assigned to a weight vector.

Since the effects of the proposed solution arrangement depend on the distribution of randomly generated solutions in the objective space, all solutions cannot be assigned to their nearest weight vectors. However, the above procedure can improve the relations between initial solution and weight vector compared with the conventional method. In the case of the example solutions and weight vectors shown in Figure 1, we experimentally verified that pairs of solution and weight becomes the relations shown in Figure 1 (b).

Table 1. MOEA/D based algorithms compared in this work

	Algorithm 1 using	
	Initialization	Solution Update
Conventional MOEA/D	Algorithm 2	Algorithm 3
Proposed MOEA/D-A	Algorithm 4	Algorithm 3
Conventional MOEA/D-CRU	Algorithm 2	Algorithm 5
Proposed MOEA/D-CRU-A	Algorithm 4	Algorithm 5

4 Experimental Settings

To verify the effects of the proposed chain-reaction solution arrangement, this work compares the search performances of four MOEA/D based algorithms. They are the conventional MOEA/D (Zhang et al., 2007) and MOEA/D-CRU (Sato, 2016) without the proposed solution arrangement and the proposed MOEA/D-A and MOEA/D-CRU-A with the proposed solution Arrangement. The differences among four algorithms are described in Table 1.

As the test problems, this work employs DTLZ4 problem framework (Deb et al., 2002a). In the DTLZ test suite, DTLZ4 is the problem framework which can control the difficulty to obtain uniformly distributed non-dominated solutions in the objective space by the problem parameter α. DTLZ4 with $\alpha = 1$ is equivalent to DTLZ2 problem, and $\alpha = 100$ is generally used problem setting for DTLZ4. This work uses 36 patterns of DTLZ4 problems combining $m = \{2, 3, 4, 5\}$ objectives and the parameters $\alpha = \{1, 5, 10, 20, 50, 100, 200, 500, 1000\}$. Also, the number of variables are set to $n = m + 10$.

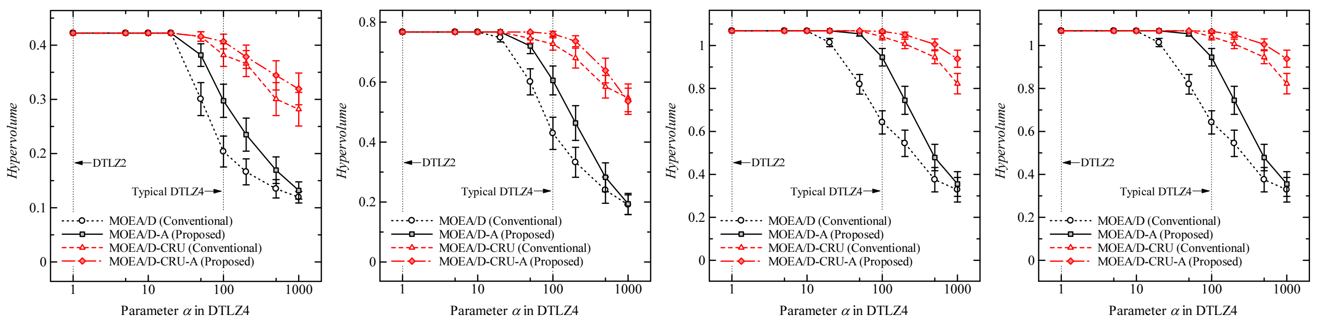
All algorithms employ the decomposition parameters $H = \{200, 19, 10, 7\}$ and the population sizes $N = \{201, 210, 286, 330\}$ for $m = \{2, 3, 4, 5\}$ objective problems, respectively. Also, the neighbor size is set to $T = 20$. To generate offspring solutions, SBX with its ratio 0.8 and distribution index 20 and the polynomial mutation with its ratio $1/n$ and distribution index 20 are used. Also, the termination condition is set to the totally 1,000 generations.

As the search performance metric, this work uses Hypervolume (HV). HV is the m-dimensional volume enclosed by the obtained non-dominated solutions and the reference point $r = (1.1, 1.1, \dots, 1.1)$ in the objective space. The higher HV, the higher search performance. In the following experiments, the average HV of 100 independent runs of each algorithm and its 95% confidence intervals are compared.

5 Results and Discussion

5.1 Search Performance at Final Generation

Figure 2 shows the average HV values obtained by the four MOEA/D based algorithms at the final generation. Figure 2 (a)-(d) are results on problems with $m =$



(a) 2 objectives (b) 3 objectives (c) 4 objectives (d) 5 objectives

Figure 2. *HV* at the final generation as the difficulty parameter α is varied

$\{2,3,4,5\}$ objectives, respectively. In each figure, the horizontal axis indicates the problem parameter α. $\alpha = 1$ is equivalent to DTLZ2, and $\alpha = 100$ is the typical DTLZ4. Also, each figure also shows the 95% confidence intervals.

First, from the results on the problem with $\alpha = 1$ which has the minimum difficulty to obtain uniformly distributed solutions in the objective space, we can see that there is no difference in *HV* values among four algorithms at the final generation. However, *HV* is gradually decreased by increasing α and the difficulty to obtain uniformly distributed solutions. Next, from the results on the problem with $\alpha = 100$, we can see that the proposed MOEA/D-A achieves higher *HV* than the conventional MOEA/D. Also, the proposed MOEA/D-CRU-A achieves higher *HV* than the conventional MOEA/D-CRU. The similar tendency can be seen on DTLZ4 problems with all objectives used in this work. Also, the proposed MOEA/D-CRU-A achieves the highest *HV* on the all problems at the final generation. These results reveal that the proposed chain-reaction solution arrangement contributes to improving the search performance of MOEA/D based algorithms, and its effectiveness becomes significant especially on the problems with a large α which has the difficulty to obtain uniformly distributed solutions in the objective space. However, there is the tendency that the effectiveness of the proposed chain-reaction arrangement is deteriorated on the problems with $\alpha = 1000$. In problems with a large α, since the distribution of randomly generated initial solutions is strongly biased in the objective space, the effect of the proposed method is weakened even if initial solutions are arranged by the proposed method.

5.2 Search Performance over Generations

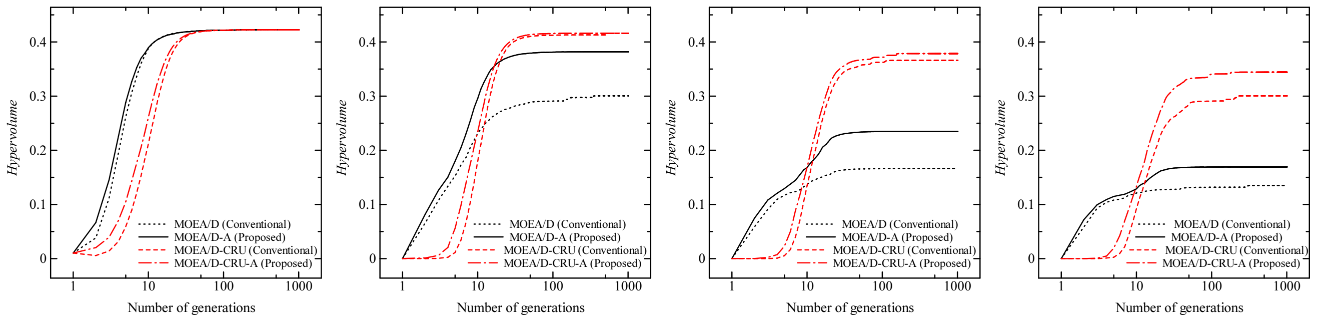
Next, we observe the transitions of the average *HV* values obtained by four algorithms over generations. Figure 3-6 shows the results on problems with 2-5 objectives, respectively. In each of them, (a)-(d) show results on problems with different α, respectively. Note that the horizontal axis is the number of generations and logarithmic scale in all figures.

From the results on problems with $\alpha = 1$ which is equivalent to DLTZ2 with the lowest difficulty, we can see that the proposed MOEA/D-A achieves higher *HV* than the conventional MOEA/D until about 50 generations. The proposed MOEA/D-CRU-A also achieves higher *HV* than the conventional MOEA/D-CRU until about 50 generations. However, after that, the difference of *HV* values of four algorithms disappears. Next, form the results on problems with $\alpha = \{50, 200, 500\}$, in almost all cases, we can see the tendency that *HV* values are saturated after about 100 generations. The proposed MOEA/D-A with the chain-reaction solution arrangement achieves higher *HV* than the conventional MOEA/D without the proposed method, however, both *HV* values are not significantly improved after about 100 generations. On the other hand, until the first 10 generations, *HV* values of MOEA/D-CRU and MOEA/D-CRU-R employing the chain-reaction solution update proposed in our previous work are lower than the ones of MOEA/D and MOEA/D-A without the chain-reaction solution update. However, the algorithms employing the chain-reaction solution update achieve higher *HV* than the algorithm without the one at the final generation. Also, we can see that MOEA/D-CRU-A shows higher *HV* than MOEA/D-CRU throughout the entire search.

These results reveal that the proposed chain-reaction solution arrangement improves the search performance of MOEA/D algorithms at any generation number.

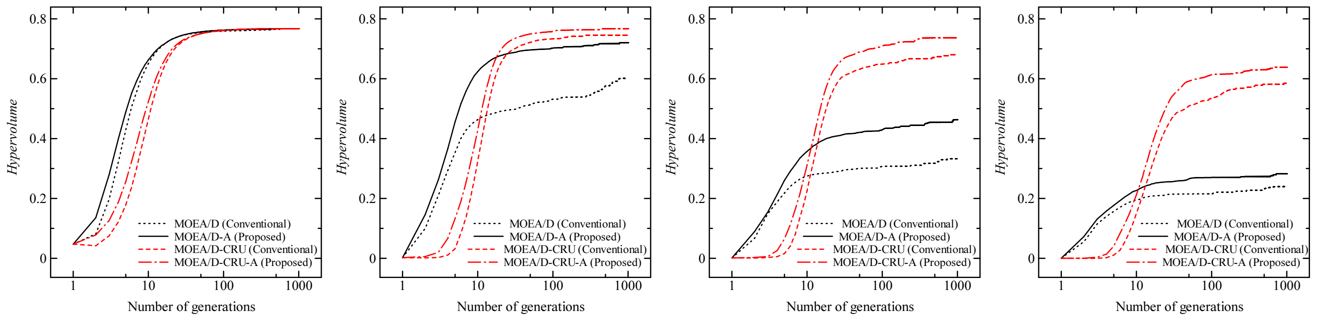
6 Conclusions

To improve the search performance of MOEA/D based algorithms on multi-objective problems by enhancing the simultaneous optimization of many single objective problems directed by weight vectors, this work proposed the chain-reaction solution arrangement method to appropriately arrange randomly generated initial solutions to weight vectors based on their positions in the objective space. The proposed method is an extension of the chain-reaction solution update and calculates the objective balance vector of each initial solutions and tries to assign it to an appropriate weight vector close the objective balance vector. The experimental results using DTLZ4 problems showed that the proposed chain-reaction solution arrangement contributes to improving the search performance of MOEA/D based algorithms.



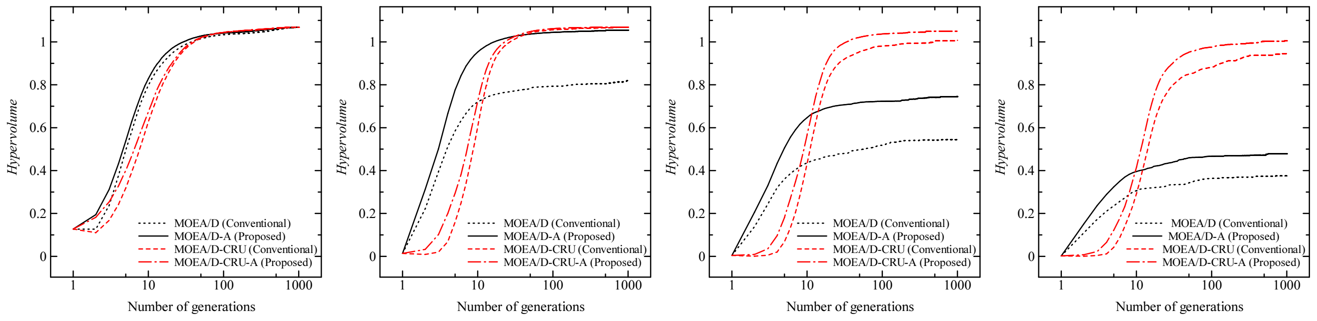
(a) $\alpha = 1$ (DTLZ2) (b) $\alpha = 50$ (c) $\alpha = 200$ (d) $\alpha = 500$

Figure 3. Transitions of HV in DTLZ4 with $m = 2$ objectives and different difficulty parameters α



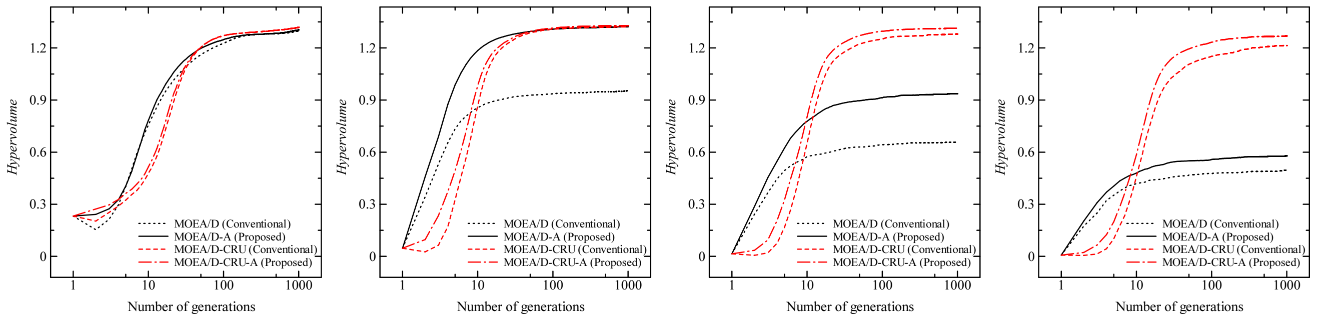
(a) $\alpha = 1$ (DTLZ2) (b) $\alpha = 50$ (c) $\alpha = 200$ (d) $\alpha = 500$

Figure 4. Transitions of HV in DTLZ4 with $m = 3$ objectives and different difficulty parameters α



(a) $\alpha = 1$ (DTLZ2) (b) $\alpha = 50$ (c) $\alpha = 200$ (d) $\alpha = 500$

Figure 5. Transitions of HV in DTLZ4 with $m = 4$ objectives and different difficulty parameters α



(a) $\alpha = 1$ (DTLZ2) (b) $\alpha = 50$ (c) $\alpha = 200$ (d) $\alpha = 500$

Figure 6. Transitions of HV in DTLZ4 with $m = 5$ objectives and different difficulty parameters α

As a future work, we will verify the effects of the proposed method on problems with many objectives and discrete solution spaces.

References

J. Bader and E. Zitzler. HypE: An Algorithm for Fast Hypervolume-based Many-objective Optimization. *Evolutionary Computation, MIT Press*, 19(1):45–76, 2011.

N. Beume, B. Naujoks, and M. Emmerich. SMS-EMOA: Multiobjective Selection Based on Dominated Hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 2007.

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

K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable Multi-objective Optimization Test Problems. *In Proc. of 2002 IEEE Congress on Evolutionary Computation (CEC2002)*, pages 825–830, 2002.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A Fast and Elitist Multi-Objective Genetic Algorithm: NSGA-II. *IEEE Trans. on Evolutionary Computation*, 6(2):182–197, 2002.

K. Deb and H. Jain. An Evolutionary Many-objective Optimization Algorithm Using Reference-point Based Non-dominated Sorting Approach, Part I: Solving Problems with Box Constraints. *IEEE Trans. on Evolutionary Computation*, 18(4): 577–601, 2014.

E. J. Hughes. Evolutionary Many-objective Optimisation: Many Once or One Many? *In Proc. of 2005 IEEE Congress on Evolutionary Computation (CEC'05)*, pp. 222–227, 2005.

K. Li, Q. Zhang, S. Kwong, M. Li, and R. Wang. Stable Matching Based Selection in Evolutionary Multiobjective Optimization. *IEEE Trans. on Evolutionary Computation*, 18(6):1–15, 2014.

H. Sato. Chain-Reaction Solution Update in MOEA/D and Its Effects on Multi and Many-Objective Optimization. *Soft Computing*, Springer, 20(10):3803–3820, 2016.

Q. Zhang and H. Li. MOEA/D: A Multi-objective Evolutionary Algorithm Based on Decomposition. *IEEE Trans. on Evolutionary Computation*, 11(6):712–731, 2007.

E. Zitzler, M. Laumanns, and L. Thiele. *SPEA2: Improving the Strength Pareto Evolutionary Algorithm. TIK-Report*, No.103, 2001.

E. Zitzler and S. Kunzili. Indicator-based Selection in Multiobjective Search. *In Proc. of the 8th Intl. Conf. on Parallel Problem Solving from Nature (PPSN VIII)*, LNCS, Vol. 3242, pages 832–842, 2004.