

Study on Applying Decentralized Evolutionary Algorithm to Asymmetric Multi-objective DCOPs with Fairness and Worst Case

Toshihiro Matsui ^a

Nagoya Institute of Technology, Gokiso-cho Showa-ku Nagoya Aichi 466-8555, Japan

Keywords: Multi-objective, Fairness, Asymmetry, Distributed Constraint Optimization, Evolutionary Algorithm, Sampling, Multiagent System.

Abstract: The Distributed Constraint Optimization Problem (DCOP) is a fundamental research area on cooperative problem solving in multiagent systems. An extended class of DCOPs represents a situation where each agent locally evaluates its partial problem with its individual constraints and objective functions on the variables shared by neighboring agents. This is a multi-objective problem on the preference of individual agents, and a set of aggregation and comparison operators is employed for a metric of social welfare among the agents. We concentrate on the case of social welfare criteria based on leximin/leximax that captures fairness among agents. Since the constraints in the practical settings of asymmetric multi-objective DCOPs are too dense for exact solution methods, scalable but inexact solution methods are necessary. We focus on employing a version of an evolutionary algorithm called AED which was designed for the original class of DCOPs. We apply the AED algorithm to asymmetric multi-objective DCOPs to handle asymmetry. We also replace the criteria in the sampling process by one of the social welfare criteria and experimentally investigate the sampling criteria in the search process.

1 INTRODUCTION

The Distributed Constraint Optimization Problem (DCOP) is a fundamental research area on cooperative problem solving in multiagent systems (Fioretto et al., 2018). With this approach, the status, the relationships and the decision making of agents are represented as a combinational optimization problem whose variables, constraints, and objective functions are partially shared by agents. A problem is solved by a decentralized optimization method that is performed by agents who exchange their information. An extended class of DCOPs represents a situation where each agent locally evaluates its partial problem with its individual constraints and objective functions on the variables shared by neighboring agents (Matsui et al., 2018a). The local evaluation is aggregated, and the assignment to the variables is globally optimized. This is a multi-objective problem on the preference of individual agents. A set of aggregation and comparison operators is employed for a metric of social welfare among the agents. We concentrate on a case of social welfare criteria based on

leximin/leximax that captures fairness among agents. Such a class of optimization problems is important for modeling practical situations where inequality among agents should be reduced. Due to the complexity of DCOPs, inexact solution methods have been employed to solve large-scale and densely constrained problems that cannot be handled by exact solution methods. Since the constraints in the practical settings of asymmetric multi-objective DCOPs are too dense for exact solution methods, scalable but inexact solution methods are necessary. In particular, opportunities are available to investigate evolutionary algorithms that are applied to multimodal problems, including multi-objective settings. Therefore, we employ a version of an evolutionary algorithm called AED, which was designed for the original class of DCOPs (Mahmud et al., 2020). We apply the AED algorithm to asymmetric multi-objective DCOPs to handle asymmetry. In addition, we replace the criteria in the sampling process by one of the social welfare criteria. We experimentally investigate the modified AED algorithms and show the effect and influence of our proposed approach.


^a  <https://orcid.org/0000-0001-8557-8167>

Table 1: Notations and parameters for AED.

P_*	set of individuals
$I, fitness$	fitness of individual I
I, x_i	assignment to variable x_i in individual I
N_i	set of neighborhood agents of agent a_i
IN	size of initial P_{a_i}
ER	coefficient for the size of several P_*
α, R_{max}	parameters for sampling of individuals
β, O_{max}	parameters for sampling of assignments to variables
MI	number of iterations between two migration steps

2 PRELIMINARIES

2.1 DCOP

A Distributed Constraint Optimization Problem (DCOP) is defined by $\langle A, X, D, F \rangle$ where A is a set of agents, X is a set of variables, D is a set of the domains of the variables, and F is a set of objective functions related to a constraint. Variable $x_i \in X$ represents the state of agent $i \in A$. Domain $D_i \in D$ is a discrete finite set of values for x_i . Objective function $f_{i,j}(x_i, x_j) \in F$ defines a cost extracted for each pair of assignments to x_i and x_j . The objective value of assignment $\{(x_i, d_i), (x_j, d_j)\}$ is defined by binary function $f_{i,j} : D_i \times D_j \rightarrow \mathbb{N}_0$. For assignment \mathcal{A} of the variables, global objective function $F(\mathcal{A})$ is defined as $F(\mathcal{A}) = \sum_{f_{i,j} \in F} f_{i,j}(\mathcal{A}_{\downarrow x_i}, \mathcal{A}_{\downarrow x_j})$, where $\mathcal{A}_{\downarrow x_i}$ is the projection of assignment \mathcal{A} on x_i .

The value of x_i is controlled by agent i , which locally knows the objective functions that are related to x_i in the initial state. The goal is to find global optimal assignment \mathcal{A}^* that minimizes the global objective value in a decentralized manner. For simplicity, we focus on the fundamental case where the scope of the functions is limited to two variables, and each agent controls a single variable.

The solution methods for DCOPs are categorized into exact and inexact solution methods (Fioretto et al., 2018). The exact methods include a few classes of algorithms based on tree search and dynamic programming (Modi et al., 2005; Petcu and Faltings, 2005). However, the complexity of such exact methods is generally exponential for several size parameters of problems. Therefore, applying exact methods to large-scale and densely constrained problems is difficult. Inexact solution methods consist of a number of approaches, including stochastic hill-climbing local search (Zhang et al., 2005) and sampling methods (Nguyen et al., 2019; Mahmud et al., 2020). We focus on the AED algorithm (Mahmud et al., 2020), which is a decentralized evolutionary algorithm for DCOPs. Their study shows that AED outperformed the other sampling-based local search methods.

```

1 Construct a BFS tree on a constraint graph.
2 Share an initial set of individuals  $P_{a_i}$  by a protocol on
  the BFS tree.
3  $Itr \leftarrow 1$ .
4 until  $Itr$  is less than a cutoff cycle do begin
5    $P_{selected} \leftarrow$  a set of  $|N_i| \times ER$  individuals sampled
     from  $P_{a_i}$  allowing to select the same elements
   .
6    $P_{new} \leftarrow \{P_{new}^{n_1}, \dots, P_{new}^{n_{|N_i|}}\}$  consisting of sets of the
     same size generated from  $P_{selected}$  by
     partitioning its elements.
7   for  $n_j$  in  $N_i$  do begin
8     Update individuals in  $P_{new}^{n_j}$  by sampling each
       assignment to  $a_i$ 's variable.
9     Send  $P_{new}^{n_j}$  to  $n_j$ .
10  end
11  for  $P_{new}^{n_{i,k}}$  received from  $n_k$  in  $N_j$  do begin
12    Update individuals in  $P_{new}^{n_{i,k}}$  by selecting each best
      assignment to  $a_i$ 's variable.
13    Return  $P_{new}^{n_{i,k}}$  to  $n_k$ .
14  end
15  for  $P_{new}^{n_j}$  returned from  $n_j$  in  $N_j$  do begin
16     $P_{a_i} \leftarrow P_{a_i} \cup P_{new}^{n_j}$ .
17  end
18   $B \leftarrow \operatorname{argmin}_{I \in P_{a_i}} I, fitness$ .
19  Update and commit the globally best solution
    using  $B$  by a protocol on the BFS tree
    executing in background.
20   $P_{a_i} \leftarrow$  a set of  $|N_i| \times ER$  individuals sampled from
     $P_{a_i}$  disallowing to select the same elements.
21  if  $Itr \bmod MI = 0$  then begin
22    for  $n_j$  in  $N_i$  do begin
23      Send a set of  $ER$  individuals, which is
        sampled from  $P_{a_i}$  disallowing to select
        the same elements, to  $n_j$ .
24    end
25    for  $P_{migrated}^{n_k}$  received from  $n_k$  in  $N_j$  do begin
26       $P_{a_i} \leftarrow P_{a_i} \cup P_{migrated}^{n_k}$ .
27    end
28  end
29   $Itr \leftarrow Itr + 1$ .
30 end

```

Figure 1: Summarized pseudo code of AED (agent a_i).

2.2 AED

The Anytime Evolutionary DCOP Algorithm (AED) (Mahmud et al., 2020) is a solution method based on an evolutionary algorithm for DCOPs. It is performed using a spanning tree on a constraint graph to aggregate solutions and determine the best assignment to the variables of agents. It is also an anytime algorithm that synchronizes the best solutions of agents in each iteration of a solution process. A breadth-first-search (BFS) tree, which is built in a preprocessing, is employed to reduce the delay in communication. We show the pseudo-code and related notations of the

AED based on a previous work (Mahmud et al., 2020) in Figure 1 and Table 1. The processing is performed in a synchronized manner. The algorithm basically maintains sets of candidate solutions (individuals) and fitness values attached to the solutions. A fitness value $I.fitness$ is defined as the total cost value for a (partial) solution.

After the construction of a BFS tree, a set of initial solutions is generated and shared by all the agents by an initialization protocol on the BFS tree. In the initialization step, each agent randomly generates a set of IN initial candidate solutions, which only contains the assignment to its own variable, and exchanges the partial solutions with all the neighborhood agents to aggregate them. Then the aggregated partial solutions are evaluated, and their partial fitness values are attached to the information of the partial solutions. The sets of partial solutions with fitness values are aggregated in a bottom-up manner based on a BFS tree. As a result, the agent of the root node obtains an initial set of global solutions with fitness values. Note that each fitness value is doubled, since two agents that are related to each constraint redundantly evaluate the same cost value. Therefore, the root agent corrects the fitness value by dividing by two. The initial set of solutions with fitness values is propagated to all the agents in a top-down manner. Initial global solutions P_{a_i} are independently updated by each agent a_i in the following phase of a decentralized synchronized process.

The main optimization part is repeated until a termination condition is satisfied. In each iteration, the following processing is performed. First, each agent generates sets of solutions P_{new}^j for each neighborhood agent a_j by sampling from its local set of solutions P_{a_i} . The sampling of solution I_j is performed with the probability $P(I_j)$ defined as

$$P(I_j) = \frac{R_j^\alpha}{\sum_{I_k \in P_{a_i}} R_k^\alpha} \quad (1)$$

$$R_j = R_{max} \times \frac{|I_{worst}.fitness - I_j.fitness| + 1}{|I_{worst}.fitness - I_{best}.fitness| + 1}, \quad (2)$$

where I_{worst} and I_{best} are the best and worst solutions in a set of solutions.

Then the agent updates the assignment to its own variable in each P_{new}^j by sampling from the variable's domain. The sampling of assignment d_i is performed with the probability $P(d_j)$ defined as

$$P(d_j) = \frac{W_{d_i}^\beta}{\sum_{d_k \in D_i} W_{d_k}^\beta} \quad (3)$$

$$W_{d_i} = O_{max} \times \frac{|O_{worst}.fitness - O_{d_i}.fitness| + 1}{|O_{worst}.fitness - O_{best}.fitness| + 1} \quad (4)$$

$$O_{d_i} = \sum_{n_k \in n_j} f_{i,k}(I.x_i, I.x_k) - \min_{d_j \in D_j} f_{i,j}(I.x_i, d_j), \quad (5)$$

where O_{worst} and O_{best} are the best and worst evaluation for all values in a variable's domain.

The update is locally performed. Note that the attached fitness values can be locally updated by adding the difference cost value of each constraint. The difference cost value δ_* is calculated as

$$\delta_* = \sum_{n_k \in N_*} f_{*,k}(I.x_*^{new}, i.x_k) - f_{*,k}(I.x_*^{old}, i.x_k). \quad (6)$$

The updated P_{new}^j is passed to neighborhood agent a_j . After all updates are received, each agent updates the assignment to its own variable in each $P_{new}^{i,k}$ received from neighborhood agent a_k . Here, the peer agent selects the best assignment to its own variable for each solutions so that it helps the update by the sender agent. The best assignment $I.x_j$ is shown as

$$I.x_j = \operatorname{argmin}_{d_j \in D_j} \sum_{n_k \in N_j} f_{j,k}(d_j, I.x_k). \quad (7)$$

Here each fitness value is also updated by Equation (6). The updated solutions are returned to a_k . Then each agent merges each P_{new}^j updated by both related agents to its own set of solutions P_{a_i} . Here the current locally best solution is updated if it is found in P_{a_i} . The best solution is propagated with a distributed snapshot algorithm performed in the background, and the assignments to the decision variables of the agents are updated to the best available solution at the same iteration. Then local set of solutions P_{a_i} is updated by sampling from itself to maintain the size of P_{a_i} .

In each MI iteration, a migration process is intermittently performed by exchanging and merging parts of the local solutions between each pair of neighboring agents. See a previous work (Mahmud et al., 2020) for details.

2.3 AMODCOP

An Asymmetric Multiple Objective DCOP on the preferences of agents (AMODCOP) (Matsui et al., 2018a) is defined by $\langle A, X, D, F \rangle$, where A , X and D are similarly defined as DCOP. Agent $i \in A$ has its local problem defined for $X_i \subseteq X$. For neighborhood agents i and j , $X_i \cap X_j \neq \emptyset$. F is a set of objective functions $f_i(X_i)$. Function $f_i(X_i) : D_{i_1} \times \dots \times D_{i_k} \rightarrow \mathbb{N}_0$ represents the objective value for agent i based on the variables in $X_i = \{x_{i_1}, \dots, x_{i_k}\}$. For simplicity, we concentrate on the case where each agent has a single variable and relates to its neighborhood agents with binary functions, which are asymmetrically defined for two related agents. Variable x_i of agent i is related to other variables by objective functions. When

x_i is related to x_j , agent i evaluates objective function $f_{i,j}(x_i, x_j)$. On the other hand, j evaluates another function $f_{j,i}(x_j, x_i)$. Each agent i has function $f_i(X_i)$ that represents the local problem of i that aggregates $f_{i,j}(x_i, x_j)$. We define the local evaluation of agent i as summation $f_i(X_i) = \sum_{j \in Nbr_i} f_{i,j}(x_i, x_j)$ for neighborhood agents $j \in Nbr_i$ related to i by objective functions.

Global objective function $\mathbf{F}(\mathcal{A})$ is defined as $[f_1(\mathcal{A}_1), \dots, f_{|A|}(\mathcal{A}_{|A|})]$ for assignment \mathcal{A} to all the variables. Here \mathcal{A}_i denotes the projection of assignment \mathcal{A} on X_i . The goal is to find assignment \mathcal{A}^* that minimizes the global objective based on a set of aggregation and evaluation structures.

2.4 Criterion of Social Welfare

Since multiple objective problems among individual agents cannot be simultaneously optimized in general cases, several criteria such as Pareto optimality are considered. However, there are generally a huge number of candidates of optimal solutions based on such criteria. Therefore, several social welfare and scalarization functions are employed. With aggregation and comparison operators \oplus and \prec , the minimization of the objectives is represented as $\mathcal{A}^* = \operatorname{argmin}_{\mathcal{A}} \oplus_{i \in A} f_i(\mathcal{A})$.

Several types of social welfare (Sen, 1997) and scalarization methods (Marler and Arora, 2004) are employed to handle objectives. In addition to the summation and comparison of scalar objective values, we consider several criteria based on the worst case objective values (Matsui et al., 2018a). Although some operators and criteria are designed for the maximization problems of utilities, we employ similar criteria for minimization problems.

Summation $\sum_{a_i \in A} f_i(X_i)$ only addresses the total cost values. Min-max criterion $\min \max_{a_i \in A} f_i(X_i)$ improves the worst case cost value. This criterion is called the Tchebycheff function. To improve the global cost values, ties on min-max are broken by comparing the summation values; the criterion is Pareto optimal (Marler and Arora, 2004).

Leximin for maximization problems is an extension of max-min, which is the maximization version of min-max. For this criterion, utility values are represented as a sorted objective vector $\mathbf{v} = \{v^1, \dots, v^{|A|}\}$ whose values are sorted in ascending order, and the comparison of two vectors is based on the dictionary order of the values in the vectors. Maximization with leximin is Pareto optimal and relatively improves the fairness among the objectives. To employ this criterion in the optimization process, partial sorted objective vectors for subsets of agents are employed,

and addition operator is generalized so that the values of two vectors are merged and resorted. See literature (Matsui et al., 2018a) for details. We employ 'leximax', which is an inverted leximin for minimization problems, where cost values are sorted in descending order. Our major goal is the optimization on leximax, while we also investigate the effect of employing other criteria in the search process.

2.5 Issue on Solution Methods

Several exact solution methods based on tree search and dynamic programming for AMODCOPs with preferences for individual agents have been proposed (Matsui et al., 2018a). However, such methods cannot be applied to large-scale and complex problems with dense constraints/functions, due to the combinatorial explosion of sub-problems. Although several approximations have been proposed for such solution methods (Matsui et al., 2018b), the accuracy of the solutions decreases when a number of constraints are eliminated from densely constrained problems. While several local search methods were addressed in earlier studies for such problems (Matsui et al., 2018b), further investigation is necessary. In a related work (Matsui, 2021), a simple stochastic local search method have been applied to AMODCOPs in which variants of a min-max criterion improve the worst case cost value among agents. However, each agent locally updates a single partial solution only considering a limited view of neighborhood agents in the search process, while a few pieces of summarized global information are shared.

On the other hand, evolutionary algorithms are often employed for multimodal problems, including multi-objective ones, in the area of centralized solvers. Therefore, we focus on AED, as a solution method for AMODCOPs with optimization criteria considering fairness. To apply the AED to this class of problems, several extensions of the data structure, optimization criteria and operators are required. Our main interest is the influence of such optimization criteria to the search process.

3 APPLYING AED TO AMODCOP WITH LEXIMAX

3.1 Handling Asymmetric Constraints and Replacing Fitness

Since the original AED algorithm is designed for symmetric DCOPs, several modifications are neces-

sary to handle asymmetric constraints. Since our interest in this study is the optimization on leximax criterion that considers the fairness among agents, we replace each fitness value $I.fitness$ by sorted cost vector $I.fitness$. The aggregation and comparison operators for sorted cost vectors are also employed.

In the initialization phase that aggregates the global solutions, a BFS tree on a constraint graph is employed. For the original DCOPs, in the aggregation of cost values based on a BFS tree, each binary constraint is redundantly evaluated by two agents related to the constraint. This doubles the aggregated cost value, and the agent of the root node in the BFS tree corrects the aggregated cost value by dividing it by two. In our case, the constraints are asymmetrically defined. Therefore, each agent independently evaluates its related constraints.

In a part of the main phase, each agent stochastically update the assignment to its own variable for all candidate solutions and locally updates the related cost values. In the original AED, the local update of the cost values can be easily performed by adding/subtracting new/old cost values of related constraints because of the symmetric constraints and aggregation with the summation operator. However, for asymmetric multi-objective problems, such an operation is impossible because the change of assignment to its own variable also changes the evaluation of the neighborhood agents. For this issue, we have to delegate the evaluation of the constraints to the agent on the opposite side. Moreover, a view of individual agents' cost values is necessary for each solution. We apply these extensions to the original algorithm and denote the additional view of agent a_i 's cost value attached to a solution I by $I.fitness_i$. The views of the cost values are generated in the initialization phase of Pa_i and maintained in the main phase when candidate solutions are updated and reevaluated.

With the extensions above, each agent locally evaluates the update of the cost values. When agent a_i evaluates its new assignment d_i^{new} in a solution I , its new local cost value is computed as

$$F_i(\mathcal{A}_i^{new}) = \sum_{n_j \in N_i} f_{i,j}(d_i^{new}, I.x_j). \quad (8)$$

The local cost value of neighborhood agent $a_j \in N_i$ is evaluated as

$$F_j(\mathcal{A}_j^{new}) = I.fitness_j + f_{j,i}(I.x_j, d_i^{new}) - f_{j,i}(I.x_j, I.x_i). \quad (9)$$

The local cost value of a_k in other agents is still $I.fitness_k$.

When agent a_i updates its own assignment in solution I , affected local cost values $I.fitness_i$ and $I.fitness_j \in N_i$ are updated. In addition, $I.fitness$ is

also updated by each new $F_k(\mathcal{A}_k^{new})$ with the following steps.

1. $I.fitness \leftarrow I.fitness \ominus I.fitness_k$
2. $I.fitness \leftarrow I.fitness \oplus F_k(\mathcal{A}_k^{new})$

Here \ominus removes an element from a sorted objective vector and \oplus inserts an element to the objective vector.

Although such views require additional resources, this is an inherent issue with this class of problems. The publication of cost values is another issue. However, to evaluate fairness or inequality among agents, some published information is necessary. In our experimental extension, each solution I redundantly has $I.fitness$ and a set of $I.fitness_i$ to avoid recalculation of sorted cost vectors, but $I.fitness$ is a compressed representation as shown in Section 3.3. While opportunities can be found to reduce the revealed information using several additional techniques, we concentrate on a search process with criteria and operators for the extended class of problems.

3.2 Sampling with Leximax Criterion

As mentioned above, our major interest is the application of sampling methods to find quasi-optimal solutions for asymmetric multi-objective problems with the criterion of fairness. We aim to improve the solution minimizing cost vectors with leximax. Therefore, the fitness value of each solution is modified to cost vectors, where its cost values are sorted in descending order. The best solution in the anytime update process is selected according to the order based on leximax. Note that other criteria, including summation and maximum value, can still be evaluated, because all the individual cost values are held in the cost vector.

Although the best solution is chosen with the leximax criterion, different criteria can be employed for the sampling operation in the search process. Since a sampling-based search has a property of local search, such criteria might be relatively efficient. In addition to the sampling operation, some techniques are necessary to compute the probability distribution for the criteria. We investigate the following criteria in the sampling.

3.2.1 Summation, Maximum Value and Augmented Tchebycheff Function

With the summation operator, the total cost value for all agents $\sum_{a_i \in A} F_i$ is employed, as in the original AED. As shown above, for asymmetric constraints, each agent evaluates the change of the cost values in opposite agents related to the agent. For this case, as a

baseline, we also evaluate a version that employs the summation to select the best solution.

Minimizing the maximum cost value for all agents $\max_{a_i \in A} F_i$ will also improve the evaluation on leximax. However, the multimodality of the cost space with an evaluation based on the maximum value can be relatively high. Therefore, as employed in multi-objective optimization, we also employ the augmented Tchebycheff function $\max_{a_i \in A} F_i + w \sum_{a_i \in A} F_i$ in which the ties of the maximum cost value are broken by the summation value. To be employed in the numeric operation, the summation value is multiplied by a sufficiently small coefficient value w and added to the maximum value.

3.2.2 Scalarized Leximax and Trimmed Leximax

Our major concern is whether sampling based on leximax has some effect in the class of evolutionary algorithms for DCOPs. To apply leximax criteria to sampling operation based on the numeric operation, scalarization of sorted objective vectors is necessary. Here we employ a simple method based on the dictionary order on sorted objective vectors (Matsui et al., 2018b). With the given minimum and maximum cost values c^\perp and c^\top , the scalar value $s(\mathbf{c}) = s(\mathbf{c})_{(|A|-1)}$ for sorted cost vector \mathbf{c} is recursively defined as $s(\mathbf{c})_{(k)} = s(\mathbf{c})_{(k-1)} \cdot (|c^\top - c^\perp| + 1) + (c_k - c^\perp)$ and $s(\mathbf{c})_{(0)} = 0$, where v_k is the k^{th} cost value in sorted vector \mathbf{c} . Note that $k = 1$ corresponds to the first cost value in a sorted cost vector. While this operation is computationally expensive, it can be performed using multi-precision variables.

Since the operation of leximin needs relatively high computational cost, we mitigate this issue by trimming the sorted cost vectors. Here, we only consider from the first to the n -th maximum values in a vector and convert them to a scalar value. In addition, ties of trimmed evaluation values can be broken using a small summation value that resembles the augmented Tchebycheff function.

3.3 Implementation and Complexity

Although the total cost values based on the summation criterion can be computed from the sorted cost vectors, we independently maintained the total cost value from the fitness values. As a maximum value, we employed the first element of each sorted cost vector. We used a run-length representation of the sorted cost vectors (Matsui et al., 2018a) implemented using a map data structure, so that the addition/subtraction of each cost value is relatively easily

performed with the indices of the cost values in the map. We employed the MPIR library for the computation with multi-precision variables to scalarize the sorted cost vectors for leximax. In this case, some part of the scalarized values might be lost in the computation of the probability for the sampling due to the underflow by casting them to the floating point type variables.

An extension with objective vectors requires a relatively large overhead to the original processing. Although the operation cost and memory usage for multiple objectives basically increase linearly with the number of objectives (i.e., the number of agents) in the worst case, there are opportunities to reduce them with additional techniques.

4 EVALUATION

We experimentally evaluated our proposed approach. We employed problems with 50 variables and c asymmetric binary constraints. The variables take values from the common domain of size $|D_i|$ for each problem setting. We evaluated with cases from the following types of cost functions.

- random: Random integer values in $[1, 100]$ based on uniform distribution.
- gamma92: Rounded random integer values in $[1, 100]$ based on gamma distribution with $\alpha = 9$ and $\beta = 2$.

We compared the following criteria for sampling.

- sum: Summation of the local cost values for all agents.
- sum-sum: ‘sum’ with the selection of the best solution based on summation. This combination is considered as a baseline.
- max: Maximum local cost value.
- maxsum: Augmented Tchebycheff function in which the ties of ‘max’ are broken with the additional summation value.
- lxm: Leximax criterion.
- tlxm3, tlxmh: Trimmed ‘lxm’ that only considers the higher part of the cost values in the sorted cost vectors. ‘tlxm3’ limits the number of cost values to three. ‘tlxmh’ employs the half of the cost values in a sorted cost vector.
- tlxm3sum: A modified ‘tlxm3’ in which the ties of ‘max’ are broken with the additional summation value.

Except for sum-sum, the best solution is selected under the leximax criterion. In addition to several criteria, we also evaluate the results with the Theil index T , which is a measurement of unfairness: $T = \frac{1}{|A|} \sum_{i=1}^{|A|} \left(\frac{f_i(X_i)}{\bar{f}} \ln \frac{f_i(X_i)}{\bar{f}} \right)$, where \bar{f} denotes the average value for all $f_i(X_i)$. T takes zero if all $f_i(X_i)$ are identical. With preliminary experiments, we set the following parameters for the AED algorithm: $IN = 5$, $ER = 5$, $\alpha = 1$, $R_{max} = 5$, $\beta = 5$, $O_{max} = 5$ and $MI = 5$. Although we set a small number of populations due to relatively expensive computation, such a setting was still effective in the original study of AED (Mahmud et al., 2020). The cut-off iteration was 1000. The results were averaged over ten trials with different initial solutions on ten problem instances for each setting.

Tables 2-5 show the final quality of the solutions at the cut-off iterations. We evaluated the summation, the maximum value and the Theil index for all the local cost values of the agents. In the case of random, $d = 3$ and $c = 250$ in Table 2, the maximum cost value was most minimized by maxsum, while the other maximax/leximax based criteria were similarly effective. Although 'sum', which selects the best solution under the leximax criterion, relatively reduced the maximum cost and the Theil index values in comparison to the sum-sum, its perturbation on the sampling process was insufficient. The leximax based criteria reduced the Theil index well. However, lxm, which exactly evaluated the criterion, did not produce the best result. There are two possible reasons for the result. The first is the influence of the locally optimal solutions in the search process. The another is that minimization on leximax does not completely correspond to optimization on the Theil index. However, tlxm3, which is an approximated version of leximax,

 Table 2: Solution quality (random, $d = 3$, $c = 250$).

Alg.	Sum.	Max.	Theil
sum	20766.4	634.4	0.0507
sum-sum	20039.0	724.9	0.0601
max	22424	573	0.0312
maxsum	22221.6	572.4	0.0331
lxm	22485.1	576.3	0.0317
tlxm3	22505.6	574.3	0.0307
tlxm3sum	22416.7	575.1	0.0320
tlxmh	22486.6	576.8	0.0317

 Table 3: Solution quality (random, $d = 5$, $c = 150$).

Alg.	Sum.	Max.	Theil
sum	10744.3	384.1	0.0989
sum-sum	10111.2	465.2	0.1236
max	12342.5	325.5	0.0476
maxsum	11913.5	321.7	0.0540
lxm	12107.9	318.6	0.0437
tlxm3	12283.1	322.2	0.0436
tlxm3sum	12216.1	321.5	0.0446
tlxmh	12142.2	319.2	0.0438

 Table 4: Solution quality (gamma92, $d = 3$, $c = 250$).

Alg.	Sum.	Max.	Theil
sum	8069.7	242.70	0.0406
sum-sum	7760.6	269.59	0.0462
max	8410.8	220.70	0.0321
maxsum	8259	220.45	0.0335
lxm	8374.2	220.90	0.0313
tlxm3	8379.9	220.30	0.0312
tlxm3sum	8382.8	220.34	0.0310
tlxmh	8373.9	220.90	0.0312

 Table 5: Solution quality (gamma92, $d = 5$, $c = 150$).

Alg.	Sum.	Max.	Theil
sum	4403.4	150.57	0.0704
sum-sum	4251.1	170.94	0.0786
max	4832.3	134.30	0.0487
maxsum	4633.3	133.19	0.0552
lxm	4757.3	132.84	0.0472
tlxm3	4792.9	133.08	0.0481
tlxm3sum	4781.7	133.11	0.0484
tlxmh	4762.5	132.83	0.0466

Table 6: Execution time (gamma92).

Alg.	Exec. time [s]	
	$d = 3, c = 250$	$d = 5, c = 150$
sum	268.6	163.5
sum-sum	268.0	158.7
max	414.7	228.6
maxsum	433.3	236.0
lxm	505.4	280.2
tlxm3	460.1	234.6
tlxm3sum	469.0	243.7
tlxmh	483.9	256.4

found the fairest solution in the results. In the case of random, $d = 5$ and $c = 150$ in Table 3, lxm found the best solution for the minimum cost value, although tlxm3 found the fairest solution, similar to the previous result.

In the results of Table 4 for a case of gamma 92, the maximum cost values by min-max/leximax based criteria are relatively similar due to the problem setting related to the density of the constraints and the gamma distribution of the cost values. On the other hand, the difference between the Theil index value of maxsum and that of leximax based criteria is relatively large. In another case of gamma 92 shown in Table 5, the result of tlxmh is best for the maximum value and the Theil index.

In these results, the maximum cost value by 'max' exceeds that of maxsum. This reveals the effect of tie-breaking with an additional comparison of the summation values. However, regarding the Theil index, the result was inverted, since the summation criterion tends to increase inequality. The results reveal the difficulty of exactly controlling the optimization performance on the maximum value and the leximax criterion. Considering the computational cost and accuracy, the trimmed version of the leximax criterion looks reasonable.

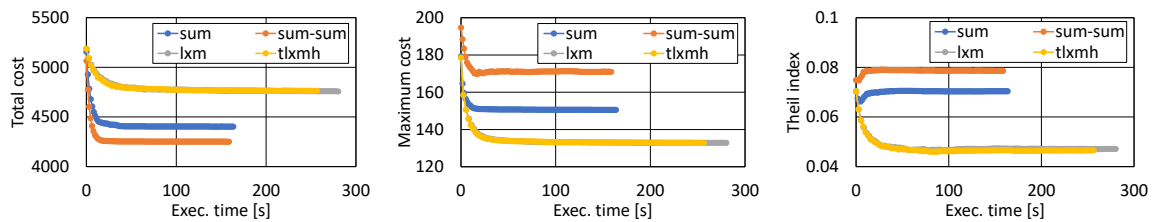
Figure 2: Anytime curve (gamma92, $d = 5$, $c = 150$).

Table 6 compares the execution time of the solution methods for 1000 iterations. The experiment was performed on a computer with g++ (GCC) 8.5.0, MPIR 3.0.0, Linux version 4.18, Intel (R) Core (TM) i9-9900 CPU @ 3.10GHz and 64GB memory. As the first study, the total computation time was evaluated excluding communication delay. Note that there are opportunities to improve our experimental implementation. A major common issue is the processing to handle the sorted objective vectors and the leximax criterion. For each criterion, the parts of the process of sampling new solution sets were differently affected by the implementation techniques. Figure 2 shows several selected anytime curves of solution quality for gamma92, $d = 5$ and $c = 150$. While the leximax-based criteria took a long execution time, their results improved in relatively earlier steps of the search process.

5 CONCLUSIONS

We applied an evolutionary algorithm called AED to asymmetric multi-objective DCOPs in which optimization is performed on a leximax criterion that improves the worst case and fairness among agents. To handle asymmetry constraints, we extended the structure in the algorithm. In addition, we replaced the criteria in the sampling process by one of social welfare criteria and experimentally investigated the sampling criteria. Our result shows the effect of sampling based on leximax-based criteria.

Our future work will include more exact evaluations using improved implementation of the algorithm, a comparison with different classes of algorithms, detailed analysis on the search space of the problems with leximax criterion, and applications to practical domains.

ACKNOWLEDGEMENTS

This work was supported in part by JSPS KAKENHI Grant Number JP19K12117.

REFERENCES

- Fioretto, F., Pontelli, E., and Yeoh, W. (2018). Distributed constraint optimization problems and applications: A survey. *Journal of Artificial Intelligence Research*, 61:623–698.
- Mahmud, S., Choudhury, M., Khan, M. M., Tran-Thanh, L., and Jennings, N. R. (2020). AED: An Anytime Evolutionary DCOP Algorithm. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 825–833.
- Marler, R. T. and Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395.
- Matsui, T. (2021). Investigation on Stochastic Local Search for Decentralized Asymmetric Multi-Objective Constraint Optimization Considering Worst Case. In *Proceedings of the 13th International Conference on Agents and Artificial Intelligence*, volume 1, pages 462–469.
- Matsui, T., Matsuo, H., Silaghi, M., Hirayama, K., and Yokoo, M. (2018a). Leximin asymmetric multiple objective distributed constraint optimization problem. *Computational Intelligence*, 34(1):49–84.
- Matsui, T., Silaghi, M., Okimoto, T., Hirayama, K., Yokoo, M., and Matsuo, H. (2018b). Leximin multiple objective dcops on factor graphs for preferences of agents. *Fundamenta Informaticae*, 158(1-3):63–91.
- Modi, P. J., Shen, W., Tambe, M., and Yokoo, M. (2005). Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180.
- Nguyen, D. T., Yeoh, W., Lau, H. C., and Zivan, R. (2019). Distributed gibbs: A linear-space sampling-based dcop algorithm. *Journal of Artificial Intelligence Research*, 64(1):705–748.
- Petcu, A. and Faltings, B. (2005). A scalable method for multiagent constraint optimization. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pages 266–271.
- Sen, A. K. (1997). *Choice, Welfare and Measurement*. Harvard University Press.
- Zhang, W., Wang, G., Xing, Z., and Wittenburg, L. (2005). Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1-2):55–87.