Interactive Algorithm for Multi-objective Constraint Optimization

Tenda Okimoto*, Yongjoon Joe*, Atsushi Iwasaki*, Toshihiro Matsui†, Katsutoshi Hirayama††, and Makoto Yokoo*

*Kyushu University, Fukuoka 8190395, Japan
†Nagoya Insutitute of Technology, Nagoya 4668555, Japan
††Kobe University, Kobe 6580022, Japan
*{tenda@agent., yongjoon@agent.,iwasaki@, yokoo@}inf.kyushu-u.ac.jp
†matsui.t@nitech.ac.jp, ††hirayama@maritime.kobe-u.ac.jp

Abstract. Many real world problems involve multiple criteria that should be considered separately and optimized simultaneously. A Multi-Objective Constraint Optimization Problem (MO-COP) is the extension of a mono-objective Constraint Optimization Problem (COP). In a MO-COP, it is required to provide the most preferred solution for a user among many optimal solutions. In this paper, we develop a novel Interactive Algorithm for MO-COP (MO-IA). The characteristics of this algorithm are as follows: (i) it can guarantee to find a Pareto solution, (ii) it narrows a region, in which Pareto front may exist, gradually, (iii) it is based on a pseudo-tree, which is a widely used graph structure in COP algorithms, and (iv) the complexity of this algorithm is determined by the induced width of problem instances. In the evaluations, we use an existing model for representing a utility function, and show empirically the effectiveness of our algorithm. Furthermore, we propose an extension of MO-IA, which can provide the more detailed information for Pareto front.

1 Introduction

Many real world optimization problems involve multiple criteria that should be considered separately and optimized simultaneously. A Multi-Objective Constraint Optimization Problem (MO-COP) [7, 8, 9, 16] is the extension of a mono-objective Constraint Optimization Problem (COP) [4, 18]. A COP is a problem to find an assignment of values to variables so that the sum of the resulting rewards is maximized. A MO-COP is a COP involves multiple criteria. In a MO-COP, generally, since trade-offs exist among objectives, there does not exist an ideal assignment, which maximizes all objectives simultaneously. Therefore, we characterize the optimal solution of a MO-COP using the concept of Pareto optimality. Solving a MO-COP is to find the Pareto front. The Pareto Front is a set of reward vectors obtained by Pareto solutions. An assignment is a Pareto solution, if there does not exist another assignment that improves all of the criteria. A COP and a MO-COP can be represented using a constraint graph, in which a node represents a variable and an edge represents a constraint.

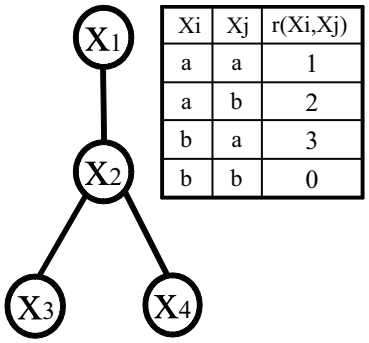
Various complete algorithms have been developed for solving a MO-COP, e.g., Russian Doll Search algorithm (MO-RDS) [17], Multi-objective AND/OR Branch-and-Bound search algorithm (MO-AOBB) [8], and MultiObjective Bucket Elimination (MO-BE) [16]. In a MO-COP, even if a constraint graph has the simplest tree structure, the size of the Pareto front, i.e., the number of Pareto solutions, is often exponential in the number of reward vectors. In such MO-COP problems, finding all Pareto solutions is not real. On the other hand, several incomplete algorithms have been developed for solving a MO-COP, e.g., Multi-Objective Mini-Bucket Elimination (MO-MBE) [16], Multi-objective Best-First AND/OR search algorithm (MO-AOBF) [9], and Multiobjective A* search algorithm (MOA*) [15]. MO-MBE computes a set of lower bounds of MO-COPs. MO-AOBF and MOA* compute a relaxed Pareto front using ϵ-dominance [14].

Various algorithms have been developed for solving a Multi-Objective Optimization Problem (MOOP) [1, 2, 3, 5, 11]. In a MOOP, a variable takes its value from a continuous domain, while a variable takes its value from a discrete domain in a MO-COP. In this paper, we focus on a MO-COP.

An Aggregate Objective Function (AOF) [12, 13] is the simplest and the most widely used classical method to find the Pareto solutions of a MOOP. This method scalarizes the set of objective functions into a weighted mono-objective function, and find an optimal solution. It is well known that an optimal solution obtained by AOF is a Pareto solution of the original MOOP problem [13]. If Pareto front is convex, AOF guarantees to find all Pareto solutions. Otherwise, it cannot find Pareto solutions in non-convex region. In our research, we use AOF to find the Pareto solutions of a MO-COP.

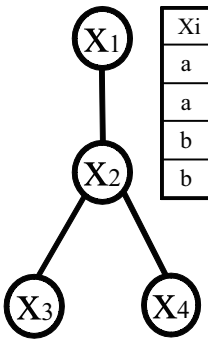
In this paper, we develop a novel Interactive Algorithm for MO-COPs (MO-IA). Our algorithm finds a set of Pareto solutions and narrows a region, in which Pareto front may exist, gradually. Our algorithm utilizes a graph structure called a pseudo-tree, which is widely used in COP algorithms. The complexity of our algorithm is determined by the induced width of problem instances. Induced width is a parameter that determines the complexity of many COP algorithms. We evaluate our algorithm using a Constraint Elasticity of Substitution (CES) utility function [10], which is widely used in many economic textbooks representing utility functions, and show empirically the effectiveness of our algorithm. Furthermore, we propose an extension of MO-IA, which finds several Pareto solutions so that we can provide a narrower region, in which Pareto front may exist, i.e., we can provide the more detailed information for Pareto front. As far as the authors aware, there exists virtually no work on interactive algorithms for a MO-COP, although various MO-COP algorithms have been developed [8, 9, 15, 16, 17].

Our proposed algorithm is similar to Physical Programming (PP) [11] and Directed Search Domain algorithm (DSD) [5]. However, these are interactive algorithms for MOOPs, while our algorithm is for MO-COPs. If we apply PP and DSD to MO-COPs, there is no guarantee to find a Pareto solution. On the other hand, our algorithm can always find a Pareto solution. Furthermore, compared to evolutionary algorithms [1, 3] for solving a MOOP, the advantage of our algorithm is that our algorithm guarantees to find a Pareto solution.



Xi	Xj	r(Xi,Xj)
a	a	1
a	b	2
b	a	3
b	b	0

Fig. 1. A mono-objective COP with four variables. The optimal solution of this problem is $\{(x_1, a), (x_2, b), (x_3, a), (x_4, a)\}$ and the optimal value is eight.



Xi	Xj	r(Xi,Xj)
a	a	(1,0)
a	b	(2,3)
b	a	(3,2)
b	b	(0,1)

Fig. 2. A bi-objective COP with four variables. The Pareto solutions of this problem are $\{\{(x_1, b), (x_2, a), (x_3, b), (x_4, b)\}, \{(x_1, a), (x_2, b), (x_3, a), (x_4, a)\}\}$, and the Pareto front is $\{(7, 8), (8, 7)\}$.

About application domains of MO-COP, we believe design/configuration tasks would be promising. For a simple toy example, we can consider Build to Order Custom Computers, where one can configure their own PC by choosing various options, e.g., CPU clock, memory size, hard drive size, operating system, and monitor size, considering multiple criteria, e.g., cost, performance, and required space.

The remainder of this paper is organized as follows. Section 2 provides some preliminaries on COPs, MO-COPs, AOF, and an existing model for representing a utility function. Section 3 introduces our interactive algorithm for MO-COPs, and Section 4 evaluates our algorithm using an existing model described in Section 2. Furthermore, we provides the extension of our algorithm. Section 5 concludes this paper and gives future works.

2 Preliminaries

In this section, we briefly describe the formalizations of Constraint Optimization Problems (COPs) and Multi-objective Constraint Optimization Problems (MO-COPs), which is the extension of a mono-objective COP. Also, we show an Aggregate Objective Function (AOF), which is the most widely used classical method to find a Pareto solution. Furthermore, we introduce a Constraint Elasticity of Substitution (CES) utility function and an indifference curve that are widely used in many economic textbooks representing utility functions.

2.1 Mono-objective Constraint Optimization Problem

A Constraint Optimization Problem (COP) [4, 18] is a problem to find an assignment of values to variables so that the sum of the resulting rewards is maximized.

A COP is defined by a set of variables X, a set of binary constraint relations C, and a set of binary reward functions F. A variable x_i takes its value from a finite, discrete domain D_i. A binary constraint relation (i, j) means there exists a constraint relation between x_i and x_j. For x_i and x_j, which have a constraint relation, the reward for an assignment $\{(x_i, d_i), (x_j, d_j)\}$ is defined by a binary reward function $r_{i,j} : D_i \times D_j \to \mathbb{R}$. For a value assignment to all variables A, let us denote

$$R(A) = \sum_{(i,j)\in C, \{(x_i,d_i),(x_j,d_j)\}\subseteq A} r_{i,j}(d_i, d_j). \tag{1}$$

Then, an optimal assignment A^* is given as $\arg\max_A R(A)$, i.e., A^* is an assignment that maximizes the sum of the value of all reward functions, and an optimal value is given by $R(A^*)$. A COP can be represented using a constraint graph, in which nodes correspond to variables and edges correspond to constraints.

A pseudo-tree is a special graph structure, which is widely used in COP algorithms. In a pseudo-tree, there exists a unique root node, and each non-root node has a parent node. The pseudo-tree contains all nodes and edges of the original constraint graph, and the edges are categorized into tree edges and back edges. There are no edges between different subtrees. For each node x_i, we denote the parent node, ancestors, and children of x_i as follows:

- p_i: the parent node, which is connected to x_i through a tree edge.
- PP_i: a set of the ancestors which are connected to x_i through back edges.
- C_i: a set of children which are connected to x_i through tree and back edges.

Example 1 (COP). Figure 1 shows a mono-objective COP with four variables x_1, x_2, x_3 and x_4. $r(x_i, x_j)$ is a binary reward function where $i < j$. Each variable takes its value assignment from a discrete domain $\{a, b\}$. The optimal solution of this problem is $\{(x_1, a), (x_2, b), (x_3, a), (x_4, a)\}$, and the optimal value is eight.

2.2 Multi-objective Constraint Optimization Problem

A Multi-Objective Constraint Optimization Problem (MO-COP) [7, 8, 9, 16] is the extension of a mono-objective COP. A MO-COP is defined by variables $X = \{x_1, \ldots, x_n\}$, multi-objective constraints $C = \{C^1, \ldots, C^m\}$, i.e., a set of sets of binary constraint relations, and multi-objective functions $O = \{O^1, \ldots, O^m\}$, i.e., a set of sets of objective functions (binary reward functions). A variable x_i takes its value from a finite, discrete domain D_i. A binary constraint relation (i, j) means there exists a constraint relation between x_i and x_j. For an objective l $(1 \leq l \leq m)$, variables x_i and x_j, which have a constraint relation, the reward for an assignment $\{(x_i, d_i), (x_j, d_j)\}$ is defined by a binary reward function $r^l_{i,j} : D_i \times D_j \to \mathbb{R}$. For an objective l and a value assignment to all variables A, let us denote

$$R^l(A) = \sum_{(i,j)\in C^l, \{(x_i,d_i),(x_j,d_j)\}\subseteq A} r^l_{i,j}(d_i, d_j). \tag{2}$$

Then, the sum of the values of all reward functions for m objectives is defined by a reward vector, denoted $R(A) = (R^1(A), \ldots, R^m(A))$. To find an assignment

that maximizes all objective functions simultaneously is ideal. However, in general, since trade-offs exist among objectives, there does not exist such an ideal assignment. Therefore, we characterize the optimal solution of a MO-COP using the concept of Pareto optimality.

Definition 1 (Dominance). *For a MO-COP and two reward vectors $R(A)$ and $R(A')$, we call that $R(A)$ dominates $R(A')$, denoted by $R(A') \prec R(A)$, iff $R(A')$ is partially less than $R(A)$, i.e., (i) it holds $R^l(A') \leq R^l(A)$ for all objectives l, and (ii) there exists at least one objective l, such that $R^l(A') < R^l(A)$.*

Definition 2 (Pareto solution). *For a MO-COP and an assignment A, we call that A is the Pareto solution, iff there does not exist another assignment A', such that $R(A) \prec R(A')$.*

Definition 3 (Pareto Front). *For a MO-COP, we call a set of reward vectors obtained by Pareto solutions as the Pareto front.*

Solving a MO-COP is to find the Pareto front. A MO-COP can be also represented using a constraint graph as a COP. In this paper, we assume that all reward values are non-negative.

Example 2 (MO-COP). Figure 2 shows a bi-objective COP, which is an extension of a mono-objective COP in Fig. 1. Each variable takes its value from a discrete domain $\{a, b\}$. The Pareto solutions of this problem are $\{\{(x_1, b), (x_2, a), (x_3, b), (x_4, b)\}, \{(x_1, a), (x_2, b), (x_3, a), (x_4, a)\}\}$, and the Pareto front is $\{(7, 8), (8, 7)\}$, which is a set of reward vectors obtained by these Pareto solutions.

2.3 Aggregate Objective Function

An Aggregate Objective Function (AOF) [12, 13] is the simplest and the most widely used classical method to find the Pareto solutions of a MOOP. This method scalarizes the set of objective functions into a weighted mono-objective function and find an optimal solution. For objective functions $o^1, \ldots, o^m$ of a MOOP, we define a weight denoted by $\alpha = (\alpha_1, \ldots, \alpha_m), where \sum_{1 \leq i \leq m} \alpha_i = 1, \alpha_i > 0$. Next, we make a weighted mono-objective function $\alpha_1 o^1 + \ldots + \alpha_m o^m$, and find the optimal solution. Then, the following theorem holds:

Theorem 1 (AOF). *For a MOOP, an optimal solution A^* obtained by AOF is a Pareto solution of the original problem.*

It is well known that AOF can guarantee to find all Pareto solutions, if Pareto front is convex. Otherwise, it cannot find all Pareto solutions. In this paper, we use this method to find the Pareto solutions of a MO-COP. Theorem 1 holds also for MO-COPs. We omit the proof due to space limitations.

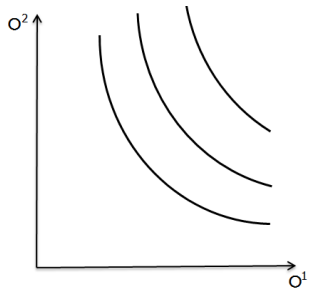


Fig. 3. Indifference curves of Cobb-Douglas function for a bi-objective COP.

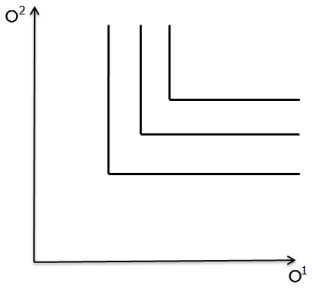


Fig. 4. Indifference curves of Leontief function for a bi-objective COP.

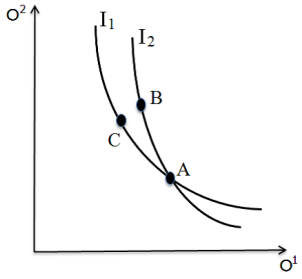


Fig. 5. Indifference curves I_1 and I_2 intersect at point A.

2.4 Constraint Elasticity of Substitution utility function

A Constraint Elasticity of Substitution (CES) utility function [10] is a function which is widely used in many economic textbooks representing utility functions. A CES utility function has the form

$$u(x_1, \ldots, x_m) = (\alpha_1 x_1^p + \ldots + \alpha_m x_m^p)^{1/p}, \tag{3}$$

where $\sum_{1 \leq i \leq m} \alpha_i = 1$, $\alpha_i > 0$, $p < 1$. Linear, Cobb-Douglas and Leontief functions are special cases of the CES utility function. For example, as $p \to 1$, the CES utility function becomes a linear function

$$u(x_1, \ldots, x_m) = \alpha_1 x_1 + \ldots + \alpha_m x_m. \tag{4}$$

As $p \to 0$, the CES utility function becomes Cobb-Douglas function

$$u(x_1, \ldots, x_m) = x_1^{\alpha_1} \times \ldots \times x_m^{\alpha_m}. \tag{5}$$

As $p \to -\infty$, the CES utility function becomes Leontief function

$$u(x_1, \ldots, x_m) = min(x_1, \ldots, x_m). \tag{6}$$

2.5 Indifference curves

The indifference curve [10, 19] shows the various combinations of goods that make a person equally satisfied. [1] For example, two different pairs, e.g., a pair of 10 compact discs and 150 candy bars, and a pair of 12 compact discs and 130 candy bars, are on the same indifference curve means that a person has a same utility, whichever pair he/she chooses.

[1] For $m \geq 3$ goods (objectives), we can consider an indifference surface.

Example 3 (Indifference curves). Figure 3 and 4 show indifference curves of Cobb-Douglas and Leontief functions for a bi-objective COP, respectively. On the graphs, o_1 represents quantity of goods, e.g., compact discs, while o_2 represents quantity of goods, e.g., candy bars. A person is equally satisfied at any point along a given curve, i.e., each point brings the same utility.

The following are the typical properties of indifference curves:

- Indifference curves are convex to the origin.
- Indifference curves cannot intersect each other.
- Higher indifference curves represents higher utility.

The first property is derived from the principle called diminishing marginal rate of substitution [19]. As a person substitutes good o_1 for good o_2, the marginal rate of substitution diminishes as o_1 for o_2 along an indifference curve. The slope of the curve is referred as the marginal rate of substitution. The marginal rate of substitution is the rate at which a person must sacrifice units of one good to obtain one more unit of another good.

We show the second property by contradiction. Assume that the indifference curves I_1 and I_2 intersect at point A (see Fig. 5). That would mean that a person is indifferent between A and all points on I_1. In particular, he/she would be indifferent between A and B, between A and C, and accordingly between B and C. However, since B involves higher values of both objective functions than C, B is clearly preferred to C. Thus, indifference curves cannot intersect each other. Furthermore, for the third property, since the combination of goods which lies on a higher indifference curve will be preferred by a person to the combination which lies on a lower indifference curve, the higher indifference curve represents a higher utility/satisfaction.

3 Interactive Algorithm for MO-COP

In this section, we develop a novel Interactive Algorithm for MO-COP (MO-IA). This algorithm finds a set of Pareto solutions and narrows a region, in which Pareto front may exist, gradually. First, we find optimal solutions of weighted mono-objective functions using AOF. Then, we provide a user with an optimal value and a region, in which Pareto front may exist. A user determines whether he/she is satisfied by the Pareto solution. If he/she is satisfied, our algorithm terminates. Otherwise, he/she chooses a preference point in the region, in which Pareto front may exist. Next, we find a point (a reward vector obtained by a Pareto solution) in the region that is closest to the user's preference point by using a distance defined in our algorithm, and update the region, in which Pareto front may exist. Then, as a new information, we provide the user with a set of Pareto solutions and a new narrower region, in which Pareto front may exist. We continue this process until the user will be satisfied by at least one of the provided Pareto solutions.

3.1 Interactive Algorithm

Our algorithm has three phases:

Phase 1 : For each objective function, find an optimal solution.
Phase 2 : For a weighted mono-objective function, find the optimal solution.
Phase 3 : Find a point in a region that is closest to a user's preference point.

Let us describe Phase 1. We use AOF to find an optimal solution for each objective function, respectively. Specifically, for m objective functions of a MO-COP, we give the following m weights $(1, 0, \ldots, 0), (0, 1, 0, \ldots, 0), \ldots, (0, \ldots, 0, 1)$ and make the m weighted objective functions $o^1, \ldots, o^m$. Then, we find an optimal solution for each weighted mono-objective function o^i $(1 \leq i \leq m)$, respectively, i.e., it is equivalent to solve m COP problems independently. In this paper, we denote the obtained m optimal values as $R^1_{max}, \ldots, R^m_{max}$.

In Phase 2, we use AOF and make a weighted mono-objective function where each weight has a same value. Then, we find the optimal solution. Specifically, for m objective functions of a MO-COP, we make the following weighted mono-objective function, denoted π, giving the weights $\alpha_1 = \frac{1}{m}, \ldots, \alpha_m = \frac{1}{m}$, and find the optimal solution.

$$\pi : \frac{1}{m}o^1 + \ldots + \frac{1}{m}o^m \tag{7}$$

Let A^* be an optimal solution of a weighted mono-objective function π. By Theorem 1, A^* is a Pareto solution of the original problem. In this paper, we call this Pareto solution as a candidate solution. For optimal values $R^1_{max}, \ldots, R^m_{max}$ obtained by Phase 1 and a candidate solution A^* obtained by Phase 2, let A be an another Pareto solution, and let $R(A)$ be a reward vector obtained by A which is different from $R(A^*)$. Then, the following theorem holds.

Theorem 2. *For reward vectors $R(A^*)$ and $R(A)$, it holds:*
(1) $\sum_{l=1}^{m} R^l(A) \leq \sum_{l=1}^{m} R^l(A^*)$.
(2) $\exists l : R^l(A^*) < R^l(A) \leq R^l_{max}$.

Proof. Since A^* is an optimal solution of a weighted mono-objective function π, it holds

$$\frac{1}{m}R^1(A) + \ldots + \frac{1}{m}R^m(A) \leq \frac{1}{m}R^1(A^*) + \ldots + \frac{1}{m}R^m(A^*). \tag{8}$$

Also, there exists no reward vector that dominates a reward vector on π.

Next, we show that it holds $\exists l : R^l(A^*) < R^l(A) \leq R^l_{max}$. Since R^l_{max} is a reward vector obtained by an optimal solution of the objective function o^l, it holds $R^l(A) \leq R^l_{max}$. Furthermore, we show that it holds $\exists l : R^l(A^*) < R^l(A)$ by contradiction. Assume that $\forall l : R^l(A^*) \geq R^l(A)$ holds. Since A^* is a candidate solution, i.e., Pareto solution, and $R(A)$ is a reward vector which is different from $R(A^*)$, there exists at least one objective l, such that $R^l(A^*) > R^l(A)$. Then, it holds $R(A) \prec R(A^*)$ by Definition 1, i.e., $R(A^*)$ dominates $R(A)$. However, since A is a Pareto solution, i.e., there exist no reward vector that dominates $R(A)$, this is a contradiction. Thus, it holds $\exists l : R^l(A^*) < R^l(A)$.

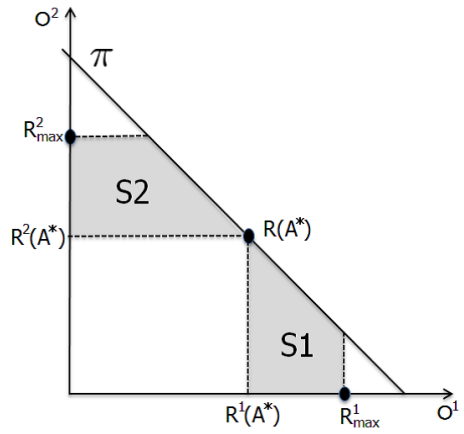


Fig. 6. A region, in which the Pareto front of a bi-objective COP may exist.

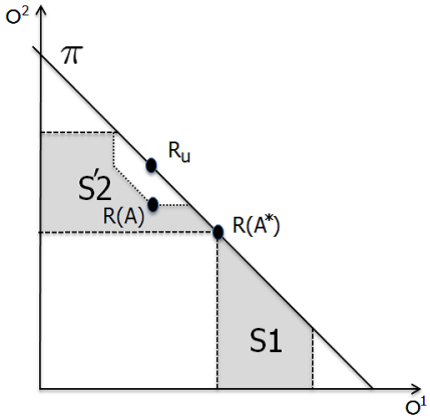


Fig. 7. A new region which is narrower compared to the region in Fig. 6.

Example 4. Figure 6 shows a region, in which the Pareto front of a bi-objective COP may exist. The x-axis represents the rewards for objective 1 and the y-axis represents those for objective 2. R^1_{max} and R^2_{max} are reward vectors obtained in Phase 1. The line π represents a weighted mono-objective function, and $R(A^*)$ is a reward vector obtained by a candidate solution A^* in Phase 2. Let $R(A)$ be a reward vector obtained by a Pareto solution, which is different from $R(A^*)$. By Theorem 2, $R(A)$ exists in the region under a function π. Furthermore, it holds $R^1(A^*) < R^1(A)$ or $R^2(A^*) < R^2(A)$. Also, since $R^1(A) \leq R^1_{max}$ and $R^2(A) \leq R^2_{max}$ must be hold, $R(A)$ exists in the region $S1$ or $S2$.

A user determines whether he/she is satisfied by a candidate solution obtained in Phase 2. If he/she is satisfied, our algorithm terminates. Otherwise, he/she chooses a point in a region, in which Pareto front may exist. We call the point as a user's preference point and denote it by $R_u (= (R^1_u, \ldots, R^m_u))$.

Let us describe Phase 3. We find a point in a region that is closest to a user's preference point. Specifically, for a reward vector $R(A)$ obtained by an assignment A and a user's preference point R_u, we define the distance between $R(A)$ and R_u as follows:

$$dis(R(A), R_u) = f(R^1(A), R^1_u) + \ldots + f(R^m(A), R^m_u),$$

$$\forall l : f(R^l(A), R^l_u) = \begin{cases} R^l_u - R^l(A) & (R^l_u \geq R^l(A)) \\ -\epsilon_l(R^l(A) - R^l_u) & (R^l_u < R^l(A)) \end{cases}, \tag{9}$$

where ϵ_l is small enough. In Phase 3, we find an assignment A so that the distance between a reward vector $R(A)$ and a user's preference point is minimal. Strictly speaking, the distance function is non-linear. However, the term $-\epsilon_l(R^l(A) - R^l_u)$ is used only for a tie-breaker. We can easily encode this metric in standard MO-COP algorithms.

Algorithm 1 MO-IA (Phase 3)

MO-IA(X,D,O)

1 Given : R_u // user's preference point on π
2 $JOIN_1$ = null,...,$JOIN_n$ = null
3 **for each** $i = n, \ldots, 1$
4 **if** i is a leaf **then**
5 $JOIN_i = R_i^{p_i} \oplus (\bigoplus_{h \in PP_i} R_i^h)$ // join all reward tables
6 Compute $\underset{a}{\operatorname{argmin}}\, dis(R(a), R_u)$ for each a in combination of assignments of p_i and PP_i
7 $JOIN_i = JOIN_i \perp_{x_i}$ // use projection to eliminate x_i
8 **else**
9 $JOIN_i = R_i^{p_i} \oplus (\bigoplus_{h \in PP_i} R_i^h) \oplus (\bigoplus_{j \in C_i} JOIN_j)$
10 Compute $\underset{a}{\operatorname{argmin}}\, dis(R(a), R_u)$ for each a in combination of assignments of p_i and PP_i
11 $JOIN_i = JOIN_i \perp_{x_i}$ // use projection to eliminate x_i
12 **end if**
13 **end for**

We show the procedure of MO-IA (Phase 3) in Algorithm 1. In our algorithm, we assume that a pseudo-tree based on total ordering $x_1, \ldots, x_n$ is given, where x_1 is a root node. The $\oplus$ operator is the operator to join two reward tables and the $\bigoplus$ operator is the operator to join all reward tables. The $\perp_x$ operator is the projection to eliminate x. $JOIN_i$ represents a reward table maintained by a node x_i. Also, $R_i^{p_i}$ and R_i^h represent the reward tables between a node x_i and its parent node p_i, and its ancestor $h \in PP_i$, respectively. Our algorithm processes bottom-up, which starts from the leaves and propagates upwards only through tree edges (line 3). If x_i is a leaf node, x_i joins all reward tables it has with its ancestor using $\bigoplus$ operator, and the reward table it has with its parent using $\oplus$ operator (line 5). Then, for each combination of assignments of p_i and PP_i, we compute an assignment so that the distance from a user's preference point is minimal (line 6). We use the $\perp$ operator to eliminate x_i from the reward table $JOIN_i$ (line 7). If x_i is not a leaf node, we access the reward tables of its children, and join the following reward tables $R_i^{p_i}$, R_i^h, and all $JOIN_{j \in C_i}$ (line 9). Then, we conduct the same process we did for a leaf node (line 10 and 11). In this algorithm, each node chooses an assignment so that the distance from a user's preference point is minimal. Thus, for an assignment to all variables A, the distance between the reward vector $R(A)$ obtained by A and a user's preference point is minimal. This is because we deal with maximization MO-COPs. We omit the proof due to space limitations. For an assignment obtained by our algorithm, the following theorem holds.

Theorem 3. *An assignment obtained by MO-IA is a Pareto solution.*

Proof. Let A^* be an assignment obtained by our algorithm and $R(A^*)$ be a reward vector obtained by A^*. We show that there exists no assignment A, such that $R(A^*) \prec R(A)$. Assume that $\exists A : R(A^*) \prec R(A)$ holds. By Definition 1, it

holds (i) $\forall l : R^l(A^*) \leq R^l(A)$ and (ii) $\exists l : R^l(A^*) < R^l(A)$. Let R_u be a user's preference point. By (i), when $R^l_u \geq R^l(A)$, the following holds for all objectives:

$$f(R^l(A), R^l_u) = R^l_u - R^l(A) \leq R^l_u - R^l(A^*) = f(R^l(A^*), R^l_u). \quad (10)$$

Otherwise, i.e., when $R^l_u < R^l(A)$, it holds

$$f(R^l(A), R^l_u) = -\epsilon_l(R^l(A) - R^l_u) \leq -\epsilon_l(R^l(A^*) - R^l_u) = f(R^l(A^*), R^l_u). \quad (11)$$

By (ii), when $R^l_u \geq R^l(A)$, the following holds at least one objective:

$$f(R^l(A), R^l_u) < f(R^l(A^*), R^l_u). \quad (12)$$

Otherwise, i.e., when $R^l_u < R^l(A)$, it holds

$$f(R^l(A), R^l_u) < f(R^l(A^*), R^l_u). \quad (13)$$

Thus, it holds $dis(R(A), R_u) < dis(R(A^*), R_u)$. However, $dis(R(A^*), R_u)$ is minimal. This is a contradiction. Thus, A^* is a Pareto solution.

In Phase 3, we can obtain a Pareto solution that gives the closest point to a user's preference point. It means that there exists no Pareto front within the distance from a user's preference point to a point obtained by Phase 3. Thus, the new region, in which the Pareto front may exist, is the remaining region obtained from the original region removing the region within this distance.

Example 5. Figure 7 shows a new region, in which the Pareto front may exist. $R(A)$ represents a reward vector obtained by our algorithm (Phase 3). Since there exists no Pareto solution within the distance $dis(R(A), R_u)$, a new region is the region obtained by removing the region enclosed by π and $S'2$ from the original region. The new region is narrower compared to that in Fig. 6.

Let a Pareto solution obtained by Phase 3 be a new candidate solution. A user determines whether he/she is satisfied by at least one of the two candidate solutions, i.e., the first candidate solution obtained by Phase 2 or a new candidate solution. If he/she is satisfied, our algorithm terminates. Otherwise, he/she chooses a new preference point in the new narrower region, in which Pareto front may exist. We conduct the Phase 3 repeatedly, i.e., we compute a set of candidate solutions and the narrowed regions, in which Pareto front may exist, until the user is satisfied by at least one of the candidate solutions. Since our algorithm repeatedly narrows the region where the Pareto front can exist, we can expect that it converges after a finite number of iterations. However, there exists a pathological case where the algorithm repeats infinitely. This happens when the user's preference is Leontief, which is very different from our distance function. To guarantee the terminating of this algorithm, we need to set a threshold value, where the user terminates the iteration when a possible maximal improvement becomes less than the threshold.

Complexity

Our algorithm MO-IA is time $O(e \times m \times |D|^{w^*+1})$ and space $O(n \times m \times |D|^{w^*})$, where n is the number of variables, m is the number of objectives, $|D|(= |D_1| = , \ldots, = |D_n|)$ is the domain size, w^* is the induced width, e is the number of constraints. The complexity of MO-IA is determined by the induced width of a problem instance. Induced width is a parameter that determines the complexity of many COP algorithms. Specially, if a problem instance has the tree structure, i.e., the induced width is one, the complexity of MO-IA is constant.

4 Evaluations

In this section, we evaluate our algorithm using CES utility functions. Specifically, we define the following four users that have different utility functions, and examine the number of the required iterations for each user until our algorithm terminates.

User 1 : Linear utility function.
User 2 : CES utility function where the parameter p is 0.5.
User 3 : Cobb-Douglas utility function.
User 4 : Leontief utility function.

Let us explain how we examine the number of the required iterations. First, we compute a candidate solution and a region, in which Pareto front may exist. Then, we determine a user's preference point, which is the intersection of a utility function and a weighted mono-objective function. If he/she is satisfied by the candidate solution, our algorithm terminates. Then, the number of the required iterations is one. Otherwise, we find a Pareto solution that gives the closest point to a user's preference point, and let this solution be a new candidate solution. Next, using indifference curves of the user, we determine a new user's preference point in the region, in which Pareto front may exist, i.e., Pareto front without the computed candidate solutions. If he/she is satisfied by at least one of the candidate solutions, our algorithm terminates, and the number of the required iterations is increased by one. We continue this process until he/she will be satisfied, and examine how many iterations are required for each user.

Let us describe termination conditions of our algorithm. For a MO-COP, a user's preference point $R_u(= (R_u^1, \ldots, R_u^m))$, and a reward vector $R(A)$ obtained by one of the candidate solutions, a user is satisfied, if the following holds:

$$\exists A : u(R_u^1, \ldots, R_u^m) \leq u(R^1(A), \ldots, R^m(A)), \tag{14}$$

i.e., termination conditions for user 1, 2, 3 and 4 are as follows:

Termination conditions for user 1

$$\exists A : \sum_{l=1}^{m} \alpha_l R_u^l \leq \sum_{l=1}^{m} \alpha_l R^l(A) \tag{15}$$

Table 1. Number of required iterations in bi-objective COPs

Nodes	User 1	User 2	User 3	User 4
10	2.5	1.6	2.1	25.3
20	2.5	2.0	1.9	17.4
30	2.6	2.2	1.8	12.8
40	2.8	2.3	1.6	13.4
50	2.8	2.5	1.7	12.9
60	2.9	2.3	1.5	11.1
70	2.9	2.3	1.5	11.9
80	2.9	2.5	1.4	11.5
90	2.8	2.5	1.2	12.5
100	2.8	2.6	1.3	10.4

Table 2. Number of required iterations in tri-objective COPs

Nodes	User 1	User 2	User 3	User 4
10	2.4	2.3	2.3	44.4
20	2.2	2.1	2.4	33.0
30	2.2	2.2	2.3	46.8
40	2.2	2.7	2.2	36.7
50	2.4	2.3	2.0	43.6
60	2.2	2.6	2.1	38.8
70	2.3	2.6	2.0	40.7
80	2.4	2.5	2.0	30.5
90	2.3	2.3	2.0	41.8
100	2.3	2.4	2.0	42.9

Termination conditions for user 2

$$\exists A : \sum_{l=1}^{m} \alpha_l \sqrt{R_u^l} \leq \sum_{l=1}^{m} \alpha_l \sqrt{R^l(A)} \tag{16}$$

Termination conditions for user 3

$$\exists A : \prod_{l=1}^{m} (R_u^l)^{\alpha_l} \leq \prod_{l=1}^{m} (R^l(A))^{\alpha_l} \tag{17}$$

Termination conditions for user 4

$$\exists A \ \forall l : R_u^l \leq R^l(A) \tag{18}$$

In our evaluations, the domain size of each variable is two, and we chose the reward value uniformly at random from the range [0,. . . ,10] for all objectives. We generate bi/tri-objective COP problem instances randomly, and determine the parameter α of CES utility functions random for each problem instance. For each objective, we generate the same constraint graph. The number of constraints is given by $|X| * |O|$, where $|X|$ and $|O|$ are the number of variables and objectives. The results represent an average of 50 problem instances. For the parameter of the distance in Phase 3, we set that ϵ_l is 0.001 for all l.

The experimental results for bi-objective COPs are summarized in Table 1. For 10 nodes, the number of required iterations for user 1, 2 and 3 are 2.5, 1.6 and 2.1, respectively. These results are almost unchanged, when the number of nodes increases. For 100 nodes, the number of required iterations for user 1, 2 and 3 are 2.8, 2.6 and 1.3, respectively. We can see that our algorithm satisfies the preferences of user 1, 2 and 3 with few iterations. We consider that this

is because the "closest" solution defined by our algorithm are almost same as the "closest" solution that user 1, 2 and 3 think. For user 4, the number of required iterations are significantly increased compared to those for other users. In Table 1, the number of required iterations are 25.3 for 10 nodes and 10.4 for 100 nodes. We consider that this is because there exists a divergence between the "closest" solution defined by our algorithm and that user 4 thinks. Furthermore, for user 4, the number of the required iterations decreases, when the number of nodes increases. The number of the required iterations 10.4 for 100 nodes are less than half of that for 10 nodes. We consider that this is because the solution space of bi-objective COPs becomes dense, when the number of nodes increases.

We confirmed the similar results for tri-objective COPs. The experimental results are summarized in Table 2. For 10 nodes, the number of required iterations for user 1, 2 and 3 are 2.4, 2.3 and 2.3, respectively. These results are almost unchanged, when the number of nodes increases. For user 4, the number of required iterations are significantly increased compared to those for other users. The number of required iterations for user 4 increases compared to those for bi-objective COPs. We consider that this is because the Pareto solutions are sparse in tri-objective COPs compared to that in bi-objective COPs. Furthermore, we do not see any direct relationship between the number of nodes and the required iterations in Table 2. We consider that this is because Pareto solutions in three dimensional tri-objective COPs are still sparse for 100 nodes, while Pareto solutions in two dimensional bi-objective COPs becomes dense.

In summary, these experimental results reveal that our algorithm is effective for user 1, 2 and 3, i.e., CES utility functions where the parameter p is between 0 and 1. However, for user 4, the number of the required iterations are significantly increased compared to those for other users. Our future works include performing more detailed analysis, e.g., examining the relationships between the size of Pareto front and the number of nodes/objectives. Furthermore, we hope to examine the performance of our algorithm based on the utilities of real people by experiments with human subjects.

Let us propose a method to reduce the number of the required iterations for a user who has Leontief utility function. In the evaluations, our algorithm required a large number of iterations for user 4. We propose the following method to improve the results for user 4. First, we estimate a coefficient α of an utility function from a user's preference point. Next, if our algorithm does not terminate in a constant number of the required iterations, we assume that a user has Leontief utility function using the estimated α. Then, we compute Pareto front repeatedly without asking a user until the required iterations converge. We examined the number of the required iterations for user 4 using this method. We used the same problem instances in section 4, i.e., 50 bi-objective COP problem instances and 50 tri-objective COP problem instances. We set a constant number of the required iterations to three. Our algorithm terminated, when the number of the required iterations was four.

Extended MO-IA

We propose an extension of our algorithm, which finds several Pareto solutions so that we can provide a narrower region, in which Pareto front may exist, i.e., more detailed information for Pareto front. When we consider an interaction in the real world, it is natural to provide several candidate solutions. Also, a narrower region is desirable. We extend the Phase 3 of our algorithm as follows.

Phase 3' : Determine additional virtual (preference) points which are different from a user's preference point, and find a Pareto solution that is closest to each point, respectively.

In our original algorithm, we find a candidate solution and provide a region, in which Pareto front may exist, gradually. On the other hand, the extended algorithm finds several candidate solutions and provides a narrower region. The narrower region is obtained from the original region removing a region within each distance between a preference point and the corresponding candidate solution. For the virtual preference points, for example, we choose the intersections (P_1 and P_2) of function π and the border of the removed region in Fig. 7.

5 Conclusions

We developed a novel interactive algorithm for a MO-COP. This algorithm finds a set of Pareto solutions and narrows a region, in which Pareto front exist, gradually. Furthermore, we showed that the complexity of our algorithm is determined by the induced width of problem instances. In the evaluations, we defined four users using a CES utility function, and examined the number of required iterations for each user. We showed empirically that our algorithm is effective for the users, who have linear and Cobb-Douglas utility functions. Finally, we proposed a method that can reduce the number of the required iterations for a user, who has Leontief utility function. Also, we proposed an extension of MO-IA, which finds several Pareto solutions so that we can provide a narrower region, in which Pareto front may exist. As future works, we intend to apply our algorithm on challenging real world problems. Furthermore, we will develop an interactive algorithm for a multi-objective DCOP, which is formalized in [6].

References

[1] K. Bringmann, T. Friedrich, F. Neumann, and M. Wagner. Approximation-guided evolutionary multi-objective optimization. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1198–1203, 2011.

[2] I. Das and J. E. Dennis. Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM Journal on Optimization*, 8(3):631–657, 1998.

[3] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evolutionary Computation*, 6(2):182–197, 2002.

[4] R. Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.
[5] T. Erfani and V. Utyuzhnikov, Sergei. Directed search domain: a method for even generation of the Pareto frontier in multiobjective optimization. *Engineering Optimization*, 43(5):467–484, 2010.
[6] F. M. D. Fave, R. Stranders, A. Rogers, and N. R. Jennings. Bounded decentralised coordination over multiple objectives. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 371–378, 2011.
[7] U. Junker. Preference-based inconsistency proving: When the failure of the best is sufficient. In *Proceedings of the 17th European Conference on Artificial Intelligence*, pages 118–122, 2006.
[8] R. Marinescu. Exploiting problem decomposition in multi-objective constraint optimization. In *Proceedings of the 15th International Conference on Principles and Practice of Constraint Programming*, pages 592–607, 2009.
[9] R. Marinescu. Best-first vs. depth-first and/or search for multi-objective constraint optimization. In *Proceedings of the 22nd IEEE International Conference on Tools with Artificial Intelligence*, pages 439–446, 2010.
[10] A. Mas-Colell, M. D. Whinston, and J. R. Green. *Microeconomic Theory*. Oxford University Press, 1995.
[11] A. Messac and C. Mattson. Generating well-distributed sets of pareto points for engineering design using physical programming. *Optimization and Engineering*, 3(4):431–450, 2002.
[12] A. Messac, C. Puemi-sukam, and E. Melachrinoudis. Aggregate objective functions and pareto frontiers: Required relationships and practical implications. *Optimization and Engineering*, 1(2):171–188, 2000.
[13] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston, 1999.
[14] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 86–92, 2000.
[15] P. Perny and O. Spanjaard. Near admissible algorithms for multiobjective search. In *Proceedings of the 18th European Conference on Artificial Intelligence*, pages 490–494, 2008.
[16] E. Rollon and J. Larrosa. Bucket elimination for multiobjective optimization problems. *Journal of Heuristics*, 12(4-5):307–328, 2006.
[17] E. Rollon and J. Larrosa. Multi-objective russian doll search. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 249–254, 2007.
[18] T. Schiex, H. Fargier, and G. Verfaillie. Valued constraint satisfaction problems: Hard and easy problems. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 631–639, 1995.
[19] J. E. Stiglitz. *Economics*. W.W.Norton & Company, 1993.