Received: February 6, 2024. Revised: March 8, 2024.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

# Conditional Preference Networks in Multiobjective Evolutionary Algorithms

**Abdulaziz Alashaikh[1]***

[1]*University of Jeddah, Saudi Arabia*
Corresponding author's Email: asalashaikh@uj.edu.sa

**Abstract:** Conditional Preference Networks (CP-nets) have emerged as a prominent and intuitive model for representing condi-tional preferences within the AI community. However, the literature lacks a concrete investigation on their applica-bility within Multiobjective Evolutionary Algorithms (MOEAs). Many optimization problems that are tackled by MOEA have a room for conditional preferences on the decision space and thus reasoning about CP-nets in MOEA is of great importance. In this work, we motivate the use of CP-nets within the MOEA framework and provide a sim-ple yet powerful approach that can be augmented during the search to return solutions that are not only optimized for the objectives but also preferred by the decision maker's CPnet. Furthermore, we consider the case where the de-cision maker has no clear idea about her preferences at the beginning of the search but willing to engage during the search to answer simple comparison questions. Our experimental results show that it is possible to return solutions that are both optimal and preferred even when the search starts with empty preference information. The obtained solution space achieved at least 23% improvement in the average penalty score.

**Keywords:** Conditional preferences, CP-nets, Multiobjective, Evolutionary algorithms, Decision space.

## 1. Introduction

In Multiobjective Optimization Problems (MOPs) [1, 2], it is quite common for the decision maker (DM) to express preferences on the space of possible solutions. Such preferences are issued with the hope of returning something desirable to the DM. Hence, taking preferences into account when solving an optimization problem is a cornerstone for the success of the approach. Evolutionary based algorithms have become the de facto method to tackle complex optimization problems. Historically, preferences in evolutionary algorithms (EAs) have been used mostly as means of reducing the size of the final Pareto Set [3-5]. While this is indeed a valid endeavor, in many problems preferences exist as part of the problem itself and expressed naturally on the decision space.

We focus on preferences as a notion that is required to be tackled and considered within MOPs. Many optimization problems have a room for (possibly conditional) preferences and thus reasoning about preferences during the search is of great importance. Two major issues need to be addressed when handling preferences in EAs. First, articulating preferences need to assume minimum efforts from the decision maker. Second, the set of returned solutions need to be sensible to the issued preferences. The first issue is mainly a representation (and learning) issue. It depends largely on the problem domain and decision maker abilities in specifying preferences. It is however desirable to assume preferences that are natural and simple to articulate. So that the DM would not be involved in a lengthy pro-cess. One of the well-known representation of preferences is utility functions.

However, it is known that such functions re-quire tedious effort from the DM to be articulated [6, 7]. Thus, a model that captures preferences from relatively natural and easy statements is required. The second issue requires efficient approaches that have the ability to assess potential solutions during the search and decide the most preferred solution given the preference in-formation. This is mainly due to the fact that optimizing a set of objectives (in the objective space) alone is not enough. The returned solutions (in the decision space) need to be desirable as well by the decision maker.

In this paper, we consider a class of preferences that is natural and easy to elicit from the DM. Such preferences are based on comparative statements of the form “I prefer z to z′ as a value for the attribute Z” or “If Z is assigned a value z then I prefer W to be w′ more than w”. The latter is known as conditional preference, and it asserts that the preference of W depends on another at-tribute’s value (namely Z). In particular, we con-sider the class of preferences that is representable by Conditional Preference Networks (CP-nets) [8]. CP-nets are intuitive graphical model to rep-resent and reason with conditional qualitative statements. Such statements were discussed by Coello [9] to be prone to the curse of dimensionality when the size of the solutions becomes big as the DM would not be able to assess her preferences over large number of alternatives. This is not an issue anymore in compact models such as CP-nets. In fact, the size of the CP-net (that the DM is required to supply) is exponentially small-er than the number of solutions it exhibits. CP-nets has been a subject of active research by AI researchers for years, but their utilization within evolutionary algorithms has not been investigated yet. The paper main contributions are: 1) We provide a simple and efficient method to incorporate CP-nets on top of any evolutionary algorithm. We also experimentally show that this method preserves the Pareto dominance and solutions diversity. 2) We discuss the scenario of interactively learning CP-net during the search based on recent advancements in the field and augment it within the search. To the best of our knowledge, this is the first attempt that proposes a general and domain-independent approach to handle user’s preferences represented by CP-nets in evolutionary algorithms.

The paper is organized as follows: a motivation example showing conditional preferences on a simple optimization problem is discussed in the next section. Section III introduces the notion of preferential dependencies as a key concept in CP nets. Formal definition of CP-nets is presented in Section IV. Section V outlines the proposed method to incorporate CP-nets in EAs. This is followed by a learning approach during the search in Section VI. Experimental results are presented in Section VII. Finally, conclusion re-marks and future work is discussed in Section VIII.

## 2. Motivating example

Consider the famous 0-1 Knapsack problem. We have a set of $n$ items where each item $x_i$ has two possible values 1 or 0 representing respectively whether the item is included in the knapsack or not. Furthermore, each item $x_i$ is associated with a value $c_i > 0$ and a weight $a_i > 0$. A knapsack has capacity $b$ and our goal is to maximize the value of the items in the knapsack.

$$\max \sum_{i=1}^{n} c_i x_i \tag{1}$$

$$\text{subject to} \sum_{i=1}^{n} a_i x_i \le b \tag{2}$$

Consider a traveler wants to pack her items. She always prefers to have her jacket in the pack in case some windchill. For the t-shirt, she prefers to have it packed only if the jacket is packed. Lastly, she prefers to have her jeans in the pack only when both the jacket and the t-shirt are in the pack. From the above information, we conclude that her preference on the jacket is unconditional, it is always preferred to have it packed. However, her preference for the t-shirt is conditioned upon whether the jacket is packed or not. Such statement shows that the preference of one attribute (i.e., T-shirt) may depend on the values of other attributes (i.e., Jacket) and they exist naturally on the decision space. One can model such preferences via CP-nets.

Informally, a CP-net is a directed graph where vertices represent attributes of the domain (jacket, t-shirt, and jeans) and every attribute has a set of possible domains (0 or 1 in our example) and edges represent preferential dependencies. Moreover, every vertex is associated with a table showing the preference function of the attribute. The CP-net that corresponds to the above traveler scenario is captured in Fig. 1. This CP-net has three attributes (or variables) every variable is annotated with a preference table. For instance, it is always preferred to pack the jacket ($1 \succ 0$ read as 1 is preferred to 0 as a value). The preference for T-shirt depends on the value of the Jacket. Thus, there is an edge from Jacket to T-shirt and its preference is $1 \succ 0$ iff Jacket is packed otherwise it is preferred not to pack the T-shirt. In such case we say that Jacket is a parent of T-shirt (similarly Jacket and T-shirt are parents for Jeans).

Given the traveler CP-net, we can determine the optimal (i.e., best) solution for this network. We simply go top to bottom in a topological order consistent with the graph and assigning every variable to its best value given the parents values. This procedure is known as sweep-forward in CP-nets [8]. Therefore, (1, 1, 1) is the best solution for the network. This is the best (i.e., most preferred) solution if we neglect the underlying optimization problem. Moreover, another important task is to

Received: February 6, 2024. Revised: March 8, 2024.

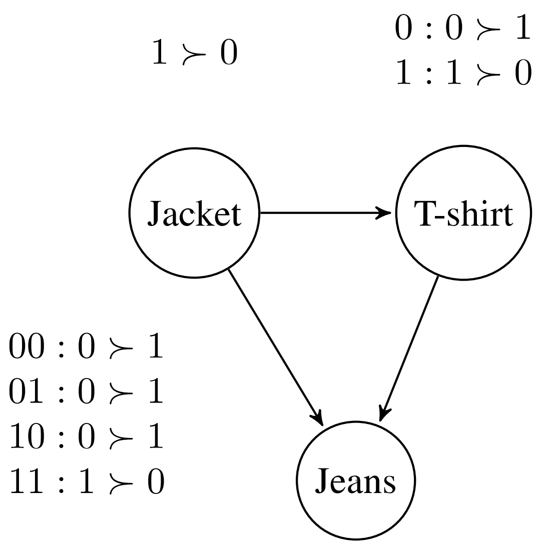


Figure. 1 The Traveler CP-net

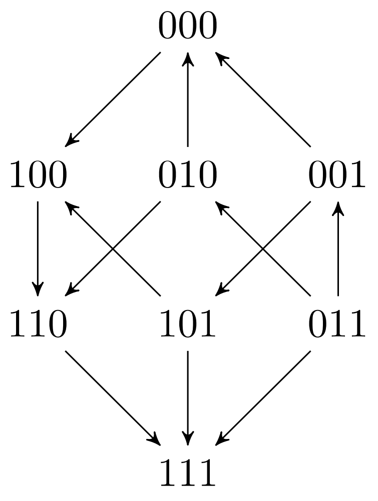


Figure. 2 Traveler Induced graph

determine whether the CP-net has enough information to prefer one solution over another. A CP-net defines a strict partial order over the set possible of solutions which determines whether one solution dominates (or preferred to) another. The full relation is defined as the transitive closure of this partial order. Fig. 2 shows the induced relation of the traveler network. Informally, a path from vertex *x* to *y* means *y* dominates *x*, similarly *y* does not dominate *x* in case there is no path from *x* to *y* in the graph. The formal definitions will be introduced in Section IV.

In the case of optimization problems, one may expect that many of the preferred solutions in the CP-net are infeasible or having poor objective values. This is indeed a possible scenario and a challenging problem. We will discuss CP-nets within the realm of MOEAs in Section V where we show one method to return solutions that are not only optimal for the optimization problem but also preferred by the decision maker's CP-net.

## 3. Preference relations and preferential independencies

In the following, we introduce basic terminologies and related concepts that would be useful in the remaining of the paper. A binary relation over a set of elements *O* is a subset of the cartesian product $O \times O$. For a given relation *R*, there are certain properties that we wish *R* to satisfy. For instance, *R* is called:

- Reflexive if and only if every element $o \in O$ is related to itself (i.e., $(o, o) \in R$).
- Irreflexive if and only if there exists no element *o* that is related to itself i.e., $(o, o) \notin R$
- Transitive if and only if for any three elements *o*, *o′*, and *o″*, if $(o, o') \in R$ and $(o', o'') \in R$ then it is the case that $(o, o'') \in R$.
- Total if and only if for any pair of elements $(o, o')$ it is the case that either $(o, o') \in R$ or $(o', o) \in R$.

**Definition 1.** A preference $\succeq$ is a binary relation that is reflexive and transitive.
$\succeq$ is also called partial preorder. The interpretation of $o \succeq o'$ is that o is as good as o′. Moreover, more precise conclusions can be derived from checking how the two elements stand with regard to each other.

1) is strictly better than *o′* (denoted as $o \succ o'$) if $o \succeq o'$ but $o' \succeq o$ does not hold.
2) is indifferent to *o′* (denoted as $o \sim o'$) if both $o \succeq o'$ and $o' \succeq o$ hold.
3) is incomparable with *o′* (denoted as $o \bowtie o'$) if both $o \not\succeq o'$ and $o' \not\succeq o$ are true.

Furthermore, a total order $\succ'$ is called a linear extension of $\succ$ if $\succ'$ is a total relation and for any two elements $o, o' \in R$ where $o \succ o'$ we have $o \succ' o'$. We work on problem domains of combinatorial nature. In such domains, every possible outcome in *O* is an assignment to a set of n attributes (or variables ) $V = \{v_i\}^n_{i=1}$. Furthermore, every $v_i$ is associated with a set of possible values $D_{vi}$ of length $m \geq 2$. For an arbitrary subset $Y = \{Y1, Y2, \ldots\} \subseteq V$ , an assignment *y* is an element of the cartesian product $DY1 \times DY2 \times \ldots$ and we refer to them as *OY* and remove *Y* from the subscript when $Y = V$ . For a specific assignment $o \in O$, we use the notation $o[Y]$ to refer the projection of *o* into the variables in *Y*.
Needless to say, the size of the possible outcomes, i.e., $|O|$ is exponential in *n*. Therefore, one cannot easily articulate a preference relation directly on the set of outcomes. Thanks to preferential dependencies, it is

possible to define the preference relation $\succeq$ in a succinct way if the given preferences exhibit some certain structure. Assume $A$ and $B$ to be any two mutual disjoint sets and ab is the result of concatenating an arbitrary assignment $a \in O_A$ and $b \in O_B$.

**Definition 2** ([8])**.** $A$ is said to be preferentially independent from another set $B$ iff $ab \succeq a'b \Leftrightarrow ab' \succeq a'b'$, for all $a, a' \in O_A$ and $b, b' \in O_B$.

In essence, $A$ is preferentially independent from another set $B$ in case the preference given to the set $A$ does not change no matter what $B$ is. That is, knowing $Y$'s values does not add anything new to the preference relation defined on the set $A$. There is also a more fine-grained conditional version of preferential dependency stated as follows.

**Definition 3** ([8])**.** Consider $A$, $B$ and $C$ to be disjoint sets of $V$ where $A \cup B \cup C = V$. We say that $A$ is independent from $B$ conditioned on another set $C$ iff $abc \succeq a'bc \Leftrightarrow ab'c \succeq a'b'c$, for all $a, a' \in O_A$, $b, b' \in O_B$ and $c \in O_C$.

We limit ourselves to work on strict (i.e., irreflexive and transitive) preference orders $\succ$. It is well-known that given any partial preorder $\succeq$, we can extract a strict order $\succ$ as follows: $o$ strictly dominates (or preferred to) $o'$ ($o \succ o'$) if and only if $o \succeq o'$ but the other way is not true. Binary relations can be visualized as directed graphs where strict orders such as $\succ$ are acyclic due to the irreflexivity property.

## 4. Conditional preference networks (CP-nets)

Informally, a CP-net is a collection of qualitative preference statements having the form: $x : y \succ y'$ which means the preference of the variable $Y$ with possible values $y$ and $y'$ is conditioned upon the value of another variable $X$. Specifically, when $X = x$, $y$ is preferred to $y'$. The preference holds only when $x$ is true. CP-nets have an appealing graphical structure to represent the preferences over $O$ in a compact way. Formally, the construction of CP-net involves two main steps:

- for every variable $v_i \in V$, the user is asked to choose "parent variables" $Pa(v_i) \subseteq V \backslash v_i$ that affect the preference relation of $v_i$.
- for every possible assignment $u \in O_{Pa(vi)}$ of $Pa(v_i)$, the user issues a total order $\succ^i_u$ over the values of $D_{vi}$.

The total order $\succ^i_u$ is known to be the statement of $v_i$ in the context of $u$. The set of all statements $\succ^i_u$ for all $u \in O_{Pa(vi)}$ is known as the conditional preference table CPT($v_i$).

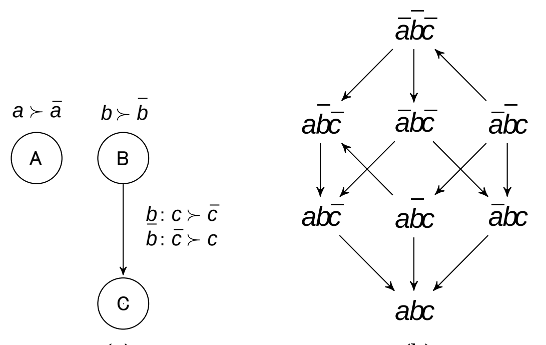


Figure. 3 An example of an acyclic CP-net with three variables and the full induced relation: (a) The model and (b) The full relation over all possible solutions (aka induced graph)

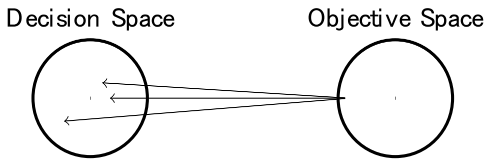


Figure. 4 One objective vector can be mapped to several decision vectors (solutions)

**Definition 4 (CP-net** [8]**).** A CP-net is defined as a pair $(G,C)$ where $G$ is a directed graph $(V, E)$ showing, for every $v \in V$, an edge from $v' \in Pa(v)$ to $v$ and $C$ is a set of CPTs for every variable in $V$.

**Example 1.** Fig. 3a shows a CP-net over $V = \{A, B, C\}$ with $D_A = \{a, \bar{a}\}$, $D_B = \{b, \bar{b}\}$, DC = $\{c, \bar{c}\}$. Each variable is annotated with its CPT. For variable $A$, the user prefers $a$ to $\bar{a}$ unconditionally. For $C$, the preference depends on the values of B, i.e., $Pa(C) = \{B\}$. For instance, in the context of $\bar{b}$, $\bar{c}$ is preferred over

We can illustrate the semantics of the CP-nets in terms of flips. A flip with respect to an outcome o result in another new outcome o′ that is different from o in exactly one variable value. The flip from $o$ to $o'$ can be either an improving flip (the new value is preferred to the old value) or worsening flip (the new value is worse compared to the old one).

Formally, let $u \in O_{Pa(vi)}$ be the parents values for a variable $v_i \in V$. Let $\succ^{v_i}_u = v^i_1 \succ \cdots \succ v^i_m$ be the preference order of $v_i$ where $u$ is the context. Then, going from $v^i_j$ to $v^i_k$ is an improving flip for $v_i$ whenever $k < j \leq m$.

**Example 2.** In Fig. 3a, $(\bar{a}\bar{b}c, \bar{a}bc)$ is an improving flip because we flipped the value of $B$ to a more preferred one. The main task in CP-nets is answering dominance queries. Such queries involve arbitrary two outcomes $o$ and $o'$ and the question is does $o$ dominate $o'$ in the underlying CPnet? The answer is yes if and only if there exists a sequence of improving flips ($\alpha_1, \alpha_2, \ldots, \alpha_k$) such that $\alpha_1 = o'$ and $\alpha_k = o$ and every pair ($\alpha_i$, $\alpha_i+1$) in this sequence is an improving flip for all $i \in \{1, \ldots, n-1\}$.

**Example 3.** Consider the network in Fig. 3a, we can conclude that $\bar{a}bc \succ \bar{a}\bar{b}\bar{c}$ as there exists a sequence of improving flips from $\bar{a}\bar{b}\bar{c}$ to $\bar{a}bc$. Specifically ($\bar{a}\bar{b}\bar{c}, \bar{a}b\bar{c}, \bar{a}bc$) is such a sequence of improving flips.

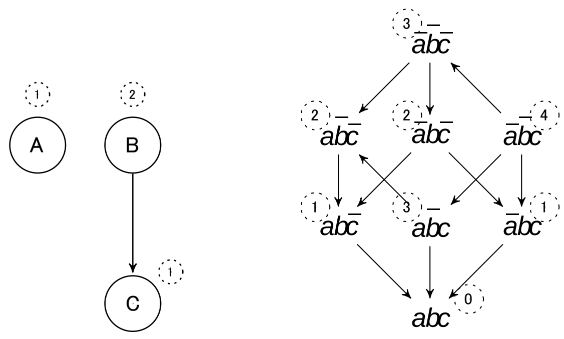
**Definition 5** (Induced Graph [8]). The induced graph for a particular CP-net $N$ is defined as a directed graph $G$ where the set of vertices is $O$ and an edge from $x$ to $y$ exists iff ($y$, $x$) form an improving flip.

The full relation of the CP-net is shown in Fig. 3b. The transitive closure of this graph is what a CP-net induces. The CP-net $N$ entails a statement $o \succ o'$ if and only if there exists a path from $o'$ to $o$ in this graph. Notice that representing the whole induced graph by requires only keeping the CP-net and its CPTs. This compact representation of CP-net is a one major advantage when working in domains with very large attributes. The complexity of answering dominance queries depends on the CP-net graph and CPTs representation. For trees (CP-nets with indegree at most one), it has shown to be linear [10] but it is PSPACE-complete for the generalized case of CP-nets [11].

## 5. Reasoning with conditional preferences in evolutionary algorithms

CP-nets are rich in semantics. In this section, we outline one possible way to augment preference information during the search. The goal is to consult the preferences specified on the decision space when searching for Pareto solutions. DMs are rarely expert in technical aspects of the problem and their preferences are usually expressed on the aspects related to potential solutions i.e., over the decision space. During the search, one objective vector could have many decision vectors or images. See e.g., Fig. 4 for an illustration. Thus, one obvious way to augment and reason with CP-nets within MOEA is to keep the non-dominated decision vectors when the objective vector has more than one mapping. In other words, we keep only the solutions that are nondominated by the underlying CP-net. This requires, for two decision vectors x and y mapped to the same objective vector $F(x) = F(y)$, determining whether $x$ is better than $y$ according to the CP-net. Such dominance question is known to be computationally hard in CP-nets and the exact complexity is known only for a restricted sub classes of CP-nets [11].

Luckily, recent advancements in reasoning with CP-net allow us to avoid dominance testing by



(a) (b).

Figure. 5 The weights and penalty values for CP-net in Example 1: (a) The variables weights and (b) The penalty for each possible solution

creating a mapping $p : O \rightarrow N$ related to the CP-net where the dominance relation can be answered in almost straightforward way i.e., if $x$ dominates $y$ then $p(x) < p(y)$ where $p(o)$ is the penalty of $o \in O$ [12]. The mapping is based on two steps. First, constructing a weight $w(v_i)$ for every variable $v_i \in V$ given the CP-net $N$. The weights represent roughly the importance of $v_i$ within the CP-net graph. Higher up nodes in the graph where many nodes depend on their values are known to be more important to be satisfied compared to others [8]. Algorithm 1 shows the steps required to create the variables weights. Second, given an outcome o, we create an indicator $d^o_{v_i} \in \{0, 1\}$ for every variable $v_i$, such that $d^o_{v_i} = 0$ iff the value of $v_i$ in the solution $o$ is the best value given its parent values $o[Pa(v_i)]$. The function $p$, or the penalty of $o$, is then defined as follows [12].

$$p(o) = \sum_{v_i \in V} w(v_i) \cdot d^o_{v_i} \tag{3}$$

Fig. 5 shows the penalty function and the variables weights for the CP-net in Fig. 3. For this small example, it is evident that whenever $p(x) < p(y)$ we either have $x \succ y$ or $x \bowtie y$ (see Corollary 1 in [12]). Given this mapping of the CP-net relation, we present a simple algorithm to integrate the CP-net within MOEAs. Essentially, we consider penalty as an objective (PaO) and use a basic non-dominated sorting (ND) technique at the heart of the algorithm (e.g., NSGA-II). Recall that, the penalty of a solution indicates the DM decision space preferences and the lower the penalty, the more preferred the solution. However, simply adding the penalty score of each solution as an additional objective would allow solutions that are far from being optimal (in the objective space sense) to appear in the Pareto Front (PF), i.e., leads to weak PF approximation. For example, if a problem has only a single solution with penalty score zero, it will always appear in the PF regardless of how bad its objective values. Ideally, penalties should not deteriorate PF optimality nor override the original objectives of the problem. Rather, they should keep the solutions quality intact to some extent but return solutions that are preferred by the DM. For this reason, some bounds on using the penalty score as objective are suggested to mitigate this undesired behavior. For example, one can utilize upper bounds on objective values to constrain the search in the objective space. Alternatively, we propose to use a penalty score conditioned upon the goodness of the solution objective values. Algorithm 2 show the approach proposed in this work where we incorporate the penalty as an additional objective only when the objective values of the solution are promising (less than the average of objective values in current population). Thus, we term the approach PaOc for constrained penalty as objective. Essentially, PaOc is just a way of reforming the problem to handle the CPnet information and can conveniently be run on top of any MOEA.

**Algorithm 1:** Calculating variables weights as in [12].

```
Input: N, an acyclic CP-net
Output: the weight $w(v_i)$ for every variable $v_i \in N$
1 Let $\pi$ be a reverse topological order for variables in
  N for $i = 1$ to $n$ do
2    if $\pi(i)$ has no children in N then
3       $w(\pi(i)) \leftarrow 1$
4    else
5       $w(\pi(i)) \leftarrow 1 + \sum_{\pi(j) \in children(\pi(i))} w(\pi(j))$
6    end
7 end
```

**Algorithm 2:** The PaOc algorithm

```
Input: a Multiobjective problem and a CP-net
Output: a Pareto Front Approximation
1 Intialize population $P$
2 Compute objective values
3 Compute $p(o)$ for each $o \in P$
4 Let $q = \sum_{s \in P} f_1(s) + \frac{f_2(s)}{|P|}$
5 foreach $o \in P$ do
6    if $f_1(o) + f_2(o) < q$ then
7       $f_3(o) = p(o)$
8    else
9       $f_3(o) = \infty$
10   end
11 end
12 while termination criterion is not met do
13   Create offspring and apply 2-8
14   Rank (based on dominance or fitness)
15 Select solutions of the next population
16 end
```

## 6. Interactive learning of CP-nets in evolutionary algorithms

So far, we have assumed the existence of a CP-net before the search starts. It is not hard to see that this assumption is unrealistic in many applications where decision makers have vague idea on their preferences especially at the beginning. Hence, decision makers need to be involved periodically during the search with the hope of revealing more information about their preferences along the way. In this section, we relax this assumption and provide an interactive approach to learn CP-nets during the search. One natural scenario is to provide the DM with a pair of solutions ($x$, $y$) and ask whether $x$ dominates $y$ (i.e., whether $x$ is more preferred to her than $y$). The answer is either "Yes" ($x$ dominates $y$) or "No" ($x$ does not dominate $y$). Fig. 6 shows the learning style in a graphical way.

The reason for choosing this learning paradigm is the fact that its cognitive burden is minimal compared to other learning styles. The users are expected to be able to assess the preference relation between two solutions instead of answering complex questions such as ranking a large set of solutions. The main challenge here, however, is learning the user's CP-net with minimum number of questions. We use recent advancements in interactive learning of CP-nets and employ optimal and near optimal learning strategies that was proposed in [13], [14]. In other words, the learning methods that we use here are guaranteed to ask the DM a near-optimal number of questions. We focus here on the two cases of separables and trees over boolean variables to simplify the presentation. The separable and tree CP-nets represent respectively CP-nets whose graph has indegree is at most zero or one. The CP-net in Fig. 1 is an example of a tree CP-net, while separables mean all variables preferences are unconditional (i.e., no edges in the graph). The general case of arbitrary acyclic CP-nets on possibly non-binary variables has been also discussed in [13], [14]. In what follows, we assume the algorithm has access to the set of variables $V$ with binary domain $D_{v_i} = \{0, 1\}$ for any $i \in \{1, 2, \ldots, n\}$ and maximum parent size $k \in \{0, 1\}$ (0 for separables and 1 for trees). We learn the CP-net by utilizing swap examples. An example is a pair of solutions ($x$, $y$) that form a question "Is $x$ better than $y$?". The example is called a swap if $x$ and $y$ differ in exactly one variable (referred to as the

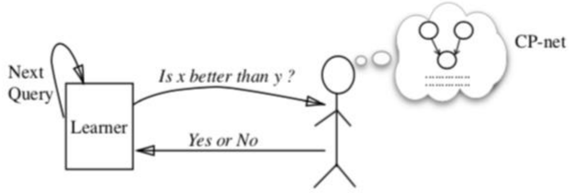


Figure. 6 Interactive learning of the decision maker's CP-net by asking simple comparative questions. The system needs to update his learned function after every query and chooses the best next query. The main challenge is to construct a sequence of questions $(x_1, y_1), (x_2, y_2) \ldots (x_r, y_r)$ that is minimum for arbitrary network

swapped variable). Let $\ell(x,y) \in \{0, 1\}$ be the answer of the DM to a question such that $\ell(x,y) = 1$ if $x$ dominates $y$ and 0 otherwise. Due to our swap assumption, answering "No" to the question means it is indeed the case that $y \succ x$ i.e., $\ell(y,x) = 1$. It is crucial to remember that for an arbitrary pair of solutions x and y, $\ell(x,y) = 0$ does not necessarily mean y dominates x ($\ell(y,x) = 1$) as both solutions can be incomparable.

## *A.* Interactive Learning of Separables

As the DM has a separable CP-net in her mind, we can learn CPT(vi) exactly by invoking the interactiveSep(x, y) procedure described in Algorithm 3, where (x, y) is an arbitrary swap example and vi is the swapped variable. Notice that swaps are naturally represented by a parent x and its offspring y that resulted from mutating x in a single bit. The learning stops as soon as we have created all the CPTs.

## *B.* Interactive Learning of Trees

When learning trees, [14] showed a near-optimal strategy and proved that $2n$ is an information theoretic lower bound for learning trees on swap examples. They devised an algorithm to learn arbitrary binary trees with $2n + e\log 2(n)$ where $e$ is the number of edges in the target tree CP-net. Algorithm 4 shows the necessary steps to interactively learn tree CP-nets. The **FindParent** procedure aims at finding the parent for the current variable's CPT. Basically, it is a binary search procedure that takes as inputs two examples on the same swapped variable v but shows conflict orders on v values (known as a conflict pair in [14]). For further information and formal proofs of the optimality for the interactive learning algorithms, interested reader is referred to [14].

Following these strategies, there could be at most $n$ interactive questions in the separable case and at most $2n + e\log_2(n)$ in the tree case where $e$ is the number of edges in the target tree CP-net. Again, during the search, the swap examples ($x$, $y$) come handy thanks to the mutation operator. However, for the tree case, one requires examples with special requirements (i.e., for each variable we need to ask two questions ($x$, $y$) and ($x'$, $y'$) where the hamming distance of $x$ and $x'$ is $n – 1$ and they are swapped for the same variable $v_i$).

```
Algorithm 3: InteractiveSep Algorithm [14]
Input: V , a set of binary variables
Output: a separable CP-net
1 foreach $v_i \in V$ do
2    find a swap example $(x,y)$ of $v_i$
3    call interactiveSep$(x, y)$
4 end
1 Procedure interactiveSep(x,y)
2    Ask the DM "Does $x$ dominates $y$"?
3    if $\ell(x,y) = 1$ then
4        CPT($v_i$) = $x[v_i] \succ y[v_i]$
5   else
6   CPT($v_i$) = $y[v_i] \succ x[v_i]$
7   end
8   return CPT($v_i$)
```

```
Algorithm 4: InteractiveTree Algorithm [14]
Input: V, a set of variables
Output: a tree CP-net
1 foreach $v_i \in V$ do
2    Let $(x, y)$ and $(x', y')$ be any two swap
     examples of $v_i$ where $x$ differs from $x'$ in all
     variables except $v_i$
3    Ask the DM "Does $x$ dominates $y$"? and
     "Does $x'$ dominates $y'$ ?"
4    if $\ell(x,y) = \ell(x',y')$ then
2        $Pa(v_i) = \emptyset$ and CPT($v_i$) =
interactiveSep$(x, y)$
3    else
4        let $P = V \setminus v_i$
5        $v_i \leftarrow$ FindParent$(P, x, y, x', y')$
6        $Pa(v_i) = v_j$
7        if $\ell(x,y) = 1$ then
8            CPT($v_i$) = $x[v_j]: x[v_i] \succ y[v_i]$
             and $x'[v_j]: y[v_i] \succ x[v_i]$
9        else
10           CPT($v_i$) = $x[v_j]: y[v_i] \succ x[v_i]$
             and $x'[v_j]: x[v_i] \succ y[v_i]$
11       end
12   end
13 end
```

```
Algorithm 5: FindParent$(P, x, y, x', y')$
Input: a set of possible parents $P \subseteq V \setminus v$ and two
swaps $(x, y)$ and $(x', y')$ of $v$ where $\ell(x,y) \neq \ell(x',y')$
Output: The parent $v_j$ of the variable $v$
1 if $|P| == 1$ then
```

```
2      return $P$
3 else
4      Let $X$ and $X'$ be a partition over $P$ with equal size
5      Let $s$ be an empty solution vector
6      foreach $v_i \in V \backslash v$ do
7         if $v_i \in X$ then
8            $s[v_i] = x[v_i]$
9         else
10           $s[v_i] = x'[v_i]$
11        end
12        Ask the DM "Does $z$ dominates $z'$"? where
            $z = s \cup x[v]$ and $z' = s \cup y[v]$
13        if $\ell(x,y) \neq \ell(z, z')$
14            FindParent($X', z, z', x, y$)
15        else
16            FindParent($X, z, z', x', y'$)
17        end
18     end
19 end
```

Given the above learning strategies, one simple idea is to learn the preference network first (by applying **interactiveSep** or **interactiveTree**) and then use the PaOc reasoning method outlined in Section 5. Another more interesting approach is to interleave learning with the search. The main idea is to progressively construct CPTs that affect the evaluation of the current generation while ignoring those that are irrelevant. Therefore, we base our evaluation on partial CP-net learned so far. Let $S$ be the set of solutions that will be used in environmental selection. For any two solutions $x$ and $y$ that form a swap example for a given $v_i \in V$, we determine CPT($v_i$) as outlined in the learning strategies. Other variables can be assumed to have no CPT and of weight zero. This partition the set of variables $V$ into two sets $V'$ and $V''$ correspond respectively to variables that have CPTs and those who doesn't. In the next generation $S'$, we do the same until $V'' = \varnothing$ and CPT($v_i$) has been determined for every $v_i \in V$. Given the population nature of MOEA, it is likely that we would be able to construct all the CPTs from few generations. This is indeed the case as shown in the experiments.

## 7. Experiments

In this section, we show the merit of our approach in preserving dominance relation in the PF while providing preferred solutions as captured by the CP-net of the DM. We ran several experiments using the settings shown in Table 1. For the CP-net generation, we used the random generator in [15]. The generator requires three parameters, namely: the number of CP-net variables $n$, maximum size on any parent set $c$, and the variable domain size (fixed to 2, i.e., Boolean variables).

As for the problems, we consider the minimization of two multiobjective problems: ZDT5 [16] and Multiobjective Knapsack (MOK) [17]. The reason for choosing those two problems is because their decision space is discrete, and the decision variables are binary. For the variables that have no preference (i.e., not part of the generated CP-net), we assume they have no effect on the preferences of the DM. We have used three algorithms as described in the table. Two variants of the proposed algorithm are considered along with a baseline algorithm; PaO

Table 1. Experiments settings

| Benchmark problem | Number of Variables | Number of Objectives | Population size | Number of Generations |
|---|---|---|---|---|
| ZDT5 problem | 50 | 2 | $N$ =100 | 100 |
| MOK problem | | | | |
| CP-net examples | Separable c0 | | Tree c1 | |
| CPnet–n..c..d2 | n: 5, 10, 20, 35 | | n: 5, 10, 20, 35 | |
| Algorithms used | • MOEA: any multiobjective evolutionary algorithm. We have adopted NSGA-II as a baseline. • MOEA-PaO: a variant of Algorithm 2 but with fixing the value of q to ∞, yielding a behavior similar to adding the penalty as a pure objective. • MOEA-PaOc: Algorithm 2. | | | |

Table 2. Penalty scores of the resulted PFs for each of the three algorithms on the ZDT5 problem averaged over 7 runs

| Algorithm | MOEA-PaO | | | MOEA-PaOc | | | MOEA | | |
|---|---|---|---|---|---|---|---|---|---|
| | min | Avg. | max | min | Avg. | max | min | Avg. | max |
| n5c0 | 0.0 | 0.1 | 2.7 | 0.0 | 0.5 | 3.0 | 0.6 | 2.5 | 4.3 |
| n5c1 | 0.0 | 0.3 | 8.3 | 0.0 | 2.4 | 10.0 | 0.7 | 8.3 | 14.3 |
| n10c0 | 0.0 | 0.7 | 7.0 | 0.0 | 1.2 | 6.0 | 2.1 | 5.2 | 8.4 |
| n10c1 | 0.0 | 1.1 | 11.6 | 0.0 | 2.9 | 13.0 | 2.7 | 10.8 | 17.4 |
| n20c0 | 0.0 | 4.0 | 11.7 | 0.0 | 5.9 | 12.0 | 5.7 | 10.6 | 16.4 |
| n20c1 | 0.0 | 7.6 | 30.4 | 2.0 | 25.4 | 45.0 | 14.4 | 40.3 | 68.0 |
| n35c0 | 0.0 | 8.6 | 16.6 | 3.0 | 9.8 | 17.0 | 13.9 | 19.1 | 24.1 |
| n35c1 | 0.4 | 27.5 | 92.7 | 6.0 | 47.9 | 113.0 | 46.9 | 97.3 | 148.4 |

Received: February 6, 2024. Revised: March 8, 2024.

Table 3. Penalty scores of the resulted PFs for each of the three algorithms on the MOK problem averaged over 7 runs

| Algorithm | MOEA-PaO | | | MOEA-PaOc | | | MOEA | | |
|---|---|---|---|---|---|---|---|---|---|
| | min | Avg. | max | min | Avg. | max | min | Avg. | max |
| n5c0 | 0.0 | 1.9 | 4.0 | 0.4 | 2.4 | 4.0 | 2.9 | 3.7 | 4.0 |
| n5c1 | 0.0 | 5.2 | 10.0 | 0.0 | 5.4 | 10.0 | 5.1 | 9.1 | 10.0 |
| n10c0 | 0.4 | 4.4 | 8.9 | 2.4 | 5.7 | 8.3 | 6.9 | 8.1 | 9.0 |
| n10c1 | 0.6 | 7.0 | 14.3 | 3.6 | 10.1 | 14.1 | 11.7 | 13.3 | 14.3 |
| n20c0 | 2.1 | 5.7 | 10.1 | 3.9 | 6.5 | 10.1 | 7.0 | 8.5 | 10.6 |
| n20c1 | 2.4 | 28.1 | 76.6 | 6.3 | 30.9 | 62.3 | 50.3 | 76.6 | 81.3 |
| n35c0 | 1.3 | 9.5 | 18.7 | 10.0 | 14.3 | 19.4 | 16.0 | 18.8 | 21.0 |
| n35c1 | 8.3 | 56.3 | 140.7 | 29.0 | 79.7 | 143.4 | 20.1 | 130.7 | 144.6 |

considers penalty as a pure objective and PaOc that considers penalty only for top ranked solutions during next population selection phase.

Each experiment is repeated 7 times and results are averaged. Tables 2 and 3 compare the penalty score of the PF achieved by each algorithm in the two problems. The results are shown in tuples which stand for minimum, average, and maximum solution penalty score. One can see from the results that both PaO approaches maintain lower penalty score than the baseline approach. This is clear for both minimum and average values. At least 23% reduction on the penalty score was achieved on averaged values. For the maximum values, however, the differences are narrower, and almost negligible for the MOK problem, as both PaO algorithms PFs would have non-dominated solutions with respect to objective values (e.g., a Pareto optimal solution with respect to objective values but not preferred by the DM and therefore has large penalty score).

Our main concern is to find optimal solutions that are also preferred as much as possible (i.e., having low penalty). In general, the possibility of having a low penalty solution is highly dependent on the problem and the CP-net structure and how they are aligned together. For example, using the PaO algorithms, it was easier to find a zero penalty solution in n5c1 than n5c0 for the MOK problem, even though the former CP-net is likely to have more complicated structure than the latter. In addition, preferred solutions are even closer to true PF in the n5c1 scenario in both algorithms.

Notice that low penalty solutions may appear in the final set of the MOEA approach, but the probability of having such a solution decreases as the CP-net size increases and without proper guidance during the search, it is unlikely that a blind MOEA

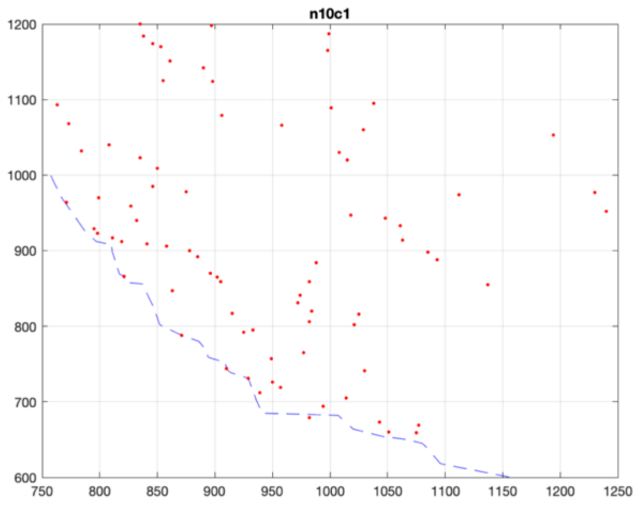


Figure. 7 The PaO algorithm with n10c1 CP-net on the MOK problem

method will encounter low penalty solutions that are also part of the PF. Indeed, it is apparent that PaO outperforms PaOc in terms of penalty score. However, in PaO there is no guarantee that preferred solutions have good objective values. For example, Fig. 7 shows the Pareto optimal solutions of the MOK problem with a random n10c1 CP-net and solved by PaO. The solutions with lowest penalty scores are far from true PF (to the upper right corner), whereas solutions with lowest objective values have comparatively larger penalties.

Figs. 8 and 9 show the results of PaOc on ZDT5 and MOK respectively compared to the true PF on different instances of CP-nets. We approximated the true PF of the MOK by basing our study on the optimal solutions returned after 1000 generations. For ZDT5, we adopted the explicit and exact true PF of the problem. Clearly, the figures show that PaOc is sufficient to provide optimal and preferred solutions.

So far the obtained results assume CP-net information is given a priori, hence we repeated the same set of experiments while CP-net is not given in advance but leaned interactively during the search. Table 3 shows the number of generations required for Algorithms 3 and 4 to learn the underlying CPnets averaged over 10 runs. In this experiment, the algorithm has access to the set of variables $V$ that form the CP-net $N$ (initially empty). Whenever it encounters a swap pair $x$ and $y$ of CPT($v_i$) not included in $N$, it learns it and updates $N$. This stops whenever there is no $v_i \in V$ with empty CPT. The environmental selection in intermediate generations is based on the learned so far network $N$ and its penalty function as described at the end of Section 6-B. The results indicate that larger CP-nets may require more generations in order to scan more alternatives for a swap pair for each variable. In terms of penalty scores and PF of the results, they are similar to PaOc results in the previous experiment.

It remains to see how PaOc performs on the diversity requirement. Fig. 10 shows the diversity of PaOc compared to the baseline MOEA algorithm for different CP-net instances. We computed diversity as follows:

$$d = \sum_{i=1}^{|P|} \sum_{j=1}^{i-1} \frac{hd(s_i, s_j)}{|P|} \tag{4}$$

where *hd* is the hamming distance between two solution vectors $s_i$ and $s_j$ and $P$ is the current population. This is known as the all-possible-pairs diversity measure [18]. From the results, we can safely conclude that PaOc is diverse to some extent.

## 8. Related work

### *A.* CP-nets and Evolutionary Algorithms

Evolutionary and Genetic algorithms have been successfully applied to different problems related to CP-nets, most notably, the problem of learning the structure of CP-net from data [19–21]. In [22], separable CP-nets have been utilized to represent the customer's preferences when placing virtual machines in cloud data centers. The placement problem is essentially an optimization problem, and the preferences were articulated apriori replacing the crowding distance in NSGA-II for the last rank. Thus, the solutions in the last rank were chosen solely based on the given separable network.

Furthermore, a weighted approximation of the CP-net has been suggested in [23] to augment uncertain preferences in the objective space where CP-net edges and nodes were associated with weights. The aim of the work is to interactively adjust the preference weights based on user's interactions in the personalized search domain.

However, all the aforementioned methods have not considered the case of articulating conditional preferences on the decision space and the problem of interactively learning the exact CP-net model when interacting with the user during the search.

### *B.* Decision Space Preferences

Typically, MOPs do not produce a single optimal solution. Instead, a set of equally good solutions known as Pareto solution are obtained or generated. Then the DM needs to select one final solution that she finds interesting. This choice commonly requires more involvement from the DM into the search process in which she expresses her preferences over the problem. Since then, researchers have continuously worked on developing algorithms that takes DM's preferences into account and search the desired area of the objective space [4, 5, 9, 24]. The aim of these algorithms is to reduce the search space, hence enable faster convergence, and satisfy DM's preferences. The preference information can take the form of trade-offs between objectives, importance of the objective functions (i.e., weights), preferred region on the objective space, or a user defined reference point ...etc. Alternatively, a utility function can be employed to represent DM's preferences in which all criteria are aggregated into a single utility function. Attributes on both decision and solution space can also be involved, on which attributes from one dimension are mapped onto the other dimension. However, it is not an easy task to generate such a representative function [25, 26].

In MOP, multiple solutions in the decision space may correspond to the same point in the objective space. When a DM chooses a Pareto solution with specific objective values, he might be interested in knowing if there are alternative solutions (pre-images) corresponding to another solution with similar objective values. Hence, having high diversity of the solutions in the decision space might be desirable. Notice that this is different from traditional objective space diversity and ensuring the latter does not necessarily imply solutions diversity [27]. Recently, an increasing attention has been paid to the diversity of the decision space in MOEA. The authors in [27] proposed a CMA-ES niching framework to the multi-criterion domain that improve decision space diversity and demonstrated that improving the diversity does not impact the convergence and diversity of the objective space. Through analyzing two real- world applications, the authors in [28] demonstrated the need to integrate special operators for diversifying solutions in the decision space in order to improve algorithm performance. An optimizer integrated into indicator-based EA that optimizes the diversity of both spaces was introduced in [29]. The algorithm allows the DM to control trade-offs between the diversity of each space. In a similar work, in [30, 31], a diversity operator is integrated into a hypervolume-based evolutionary algorithm. Furthermore, the work in [32] analyzed the solution space diversity and different crossover operators applied to instances of the binary knapsack problem and solved using different MOEA algorithms. Most notably, solution diversity is attained when two-point crossover is used. Also, NSGA-II and MSOPS evolve solutions with better diversity in both spaces. In addition, another research [33] showed that for some benchmarks problems diversity can help improving the performance of the solution. They proposed

MOEA/D with Enhanced Variable-Space Diversity (MOEA/D-EVSD) to solve these problems with higher quality solutions. Lastly, the work in [34] proposed and algorithm, vNSGA-III, that performs exploration in both spaces and improve the distribution of solutions without degrading the quality in the objective space.

However, all the previous approaches have not adopted a formal and graphical model to represent conditional preferences in the decision space. Solution diversity in the decision space indicates that multiple alternative designs, settings, options, or choices with equivalent/comparable objective quality may be available for the DM to choose from. While there has been considerable effort in exploring and exploiting diversity in decision space, yet, and to the best of our knowledge, no work has studied the incorporation of preferences represented as CP- nets on evolutionary algorithms.

## 9. Conclusions and future work

CP-nets have emerged as a natural representation for conditional preferences. This work addressed the issue of incorporating CP-nets within evolutionary algorithms, and proposed a method known as PaOc to return solutions that are not only optimal with respect to the objective values, but also preferred by the user. We also considered the case where there is no enough preference information apriori but the user is willing to answer simple pairwise questions to elicit the CP-net during the search. Our experimental results showed that PaOc was indeed a good approach for handling CP-nets within evolutionary algorithms. The proposed algorithm has achieved lower penalty scores in all tested instances compared to the baseline MOEA algorithm (a minimum of 23% improvement on average), without degrading the quality of the solution space in terms of optimality. PaOc is agnostic to the adopted strategy and can be used on top of any evolutionary algorithm.

This work aimed mainly at introducing CP-net, as an effective tool to represent and reason with conditional qualitative preferences, to the evolutionary algorithms community. However, much remains to be done. First, our work focused on discrete optimization problems. A further studies on continuous optimization problems are needed. Second, more complex integration methods for CP-nets within MOEAs may be more efficient than our PaOc method. Third, interactively learning CP-nets without the swap examples assumption may render the problem of interactive learning more interesting. Lastly, while we have focused on the decision space preferences, there is nothing limit the use CP-net on the objective space. An interesting problem in this direction is using CP-net as a selection strategy for objectives in many objectives evolutionary algorithms. In such scenario, one may adopt variants of CP-net, such as Hierarchical CP-net [19], that are capable of expressing preferences on attributes with continuous domains or discretize the domain into a finite set of values.

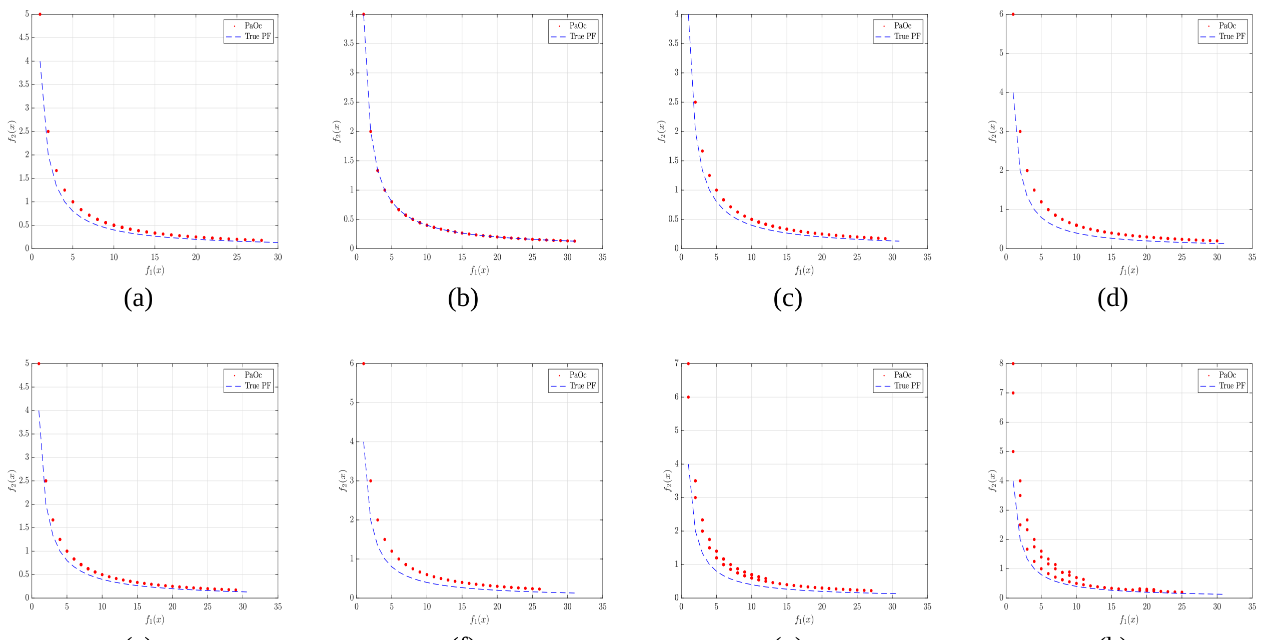


Figure. 8 The result of PaOc (red) compared to the true PF (blue) for ZDT5 after 100 generations: (a) n5c0, (b) n5c1, (c) n10c0, (d) n10c1, (e) n20c0, (f) n20c1, (g) n35c0, and (h) n35c1

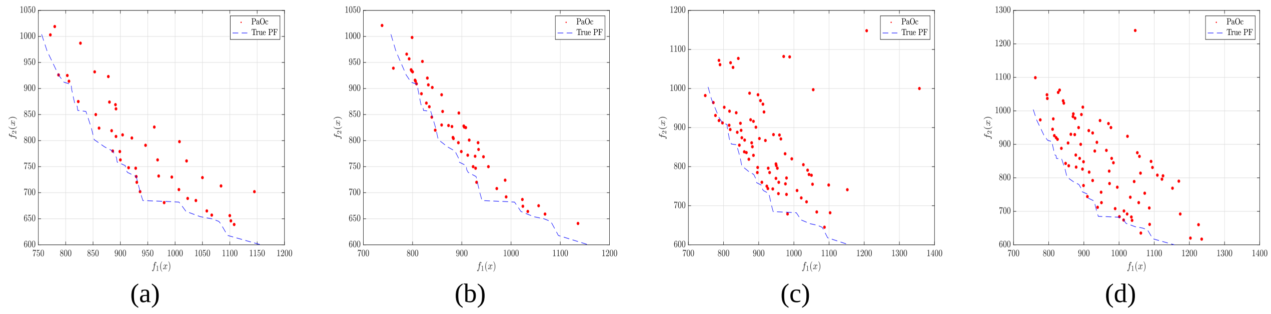
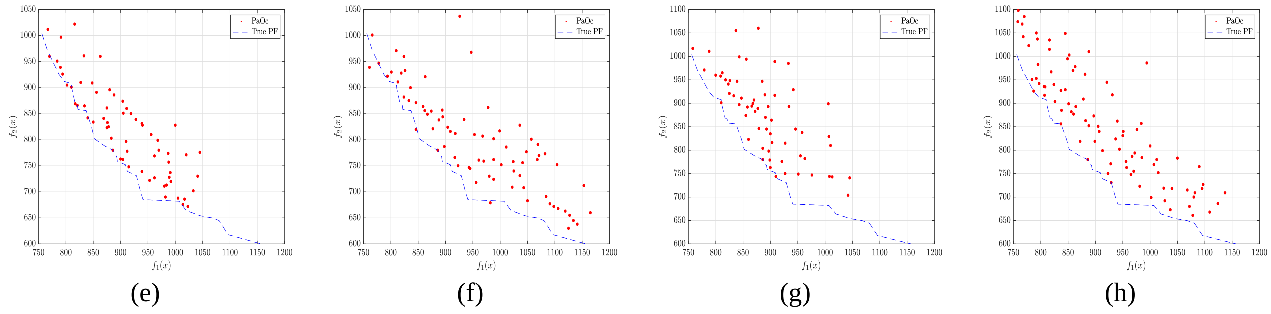



Figure. 9 The result of PaOc (red) compared to the true PF (blue) of MOK after 100 generations: (a) n5c0, (b) n5c1, (c) n10c0, (d) n10c1, (e) n20c0, (f) n20c1, (g) n35c0, and (h) n35c1

Table 4. Average number of generations required to learn the underlying CP-net

| CP-net | n5c0 | n5c1 | n10c0 | n10c1 | n20c0 | n20c1 | n35c0 | n35c1 |
|---|---|---|---|---|---|---|---|---|
| ZDT5 | 1 | 1 | 1 | 1 | 3.5 | 4.25 | 8.5 | 8.25 |
| MOK | 1 | 1 | 1 | 1 | 3.82 | 3.15 | 8.75 | 9.5 |

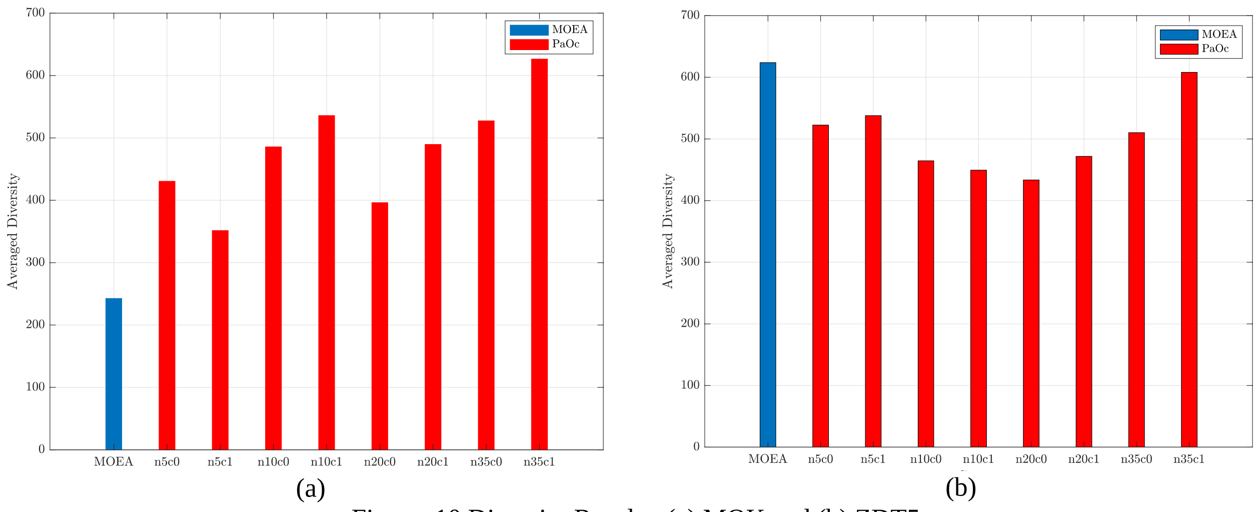


Figure. 10 Diversity Results: (a) MOK and (b) ZDT5

## Conflicts of Interest

The authors declare no conflict of interest.

## References

[1] B. Chen, W. Zeng, Y. Lin, and D. Zhang, "A new local search-based multiobjective optimization algorithm", *IEEE Transactions on Evolutionary Computation*, Vol. 19, No. 1, pp. 50–73, 2015.

[2] M. Gong, F. Liu, W. Zhang, L. Jiao, and Q. Zhang, "Interactive MOEA/D for multi-objective decision making", *Genetic and Evolutionary Computation Conference*, GECCO'11, No. 2, pp. 721–728, 2011.

[3] L. Li, Y. Wang, H. Trautmann, N. Jing, and M. Emmerich, "Multiobjective evolutionary algorithms based on target region preferences", *Swarm and Evolutionary Computation*, Vol. 40, No. 5, pp. 196–215, 2018.

[4] S. Bechikh, M. Kessentini, L. B. Said, A. R. Hurson, Ed. Elsevier, and K. Ghe´dira, "Chapter Four - Preference Incorporation in Evolutionary Multiobjective Optimization: A Survey of the State-of-the-Art", *Advances in Computers, ser. Advances in Computers*, Vol. 98, pp. 141–207, 2015.

[5] H. Wang, M. Olhofer, and Y. Jin, "A mini-review on preference modeling and articulation in multi-objective optimization: current status and challenges", *Complex & Intelligent Systems*, Vol. 3, No. 4, pp. 233– 245, 2017.

[6] J. Lang and J. Mengin, "The complexity of learning separable ceteris paribus preferences", In: *Proc. of the 21st International Joint Conference on Artificial Intelligence*, Pasadena, California, USA, pp. 848–853, 2009.

[7] R. Keeney and H. Raiffa, "*Decisions with multiple objectives: Preferences and value tradeoffs. J. Wiley*", New York, 1976.

[8] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole, "CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements," *Journal of Artificial Intelligence Research (JAIR)*, Vol. 21, pp. 135–191, 2004.

[9] C. Coello, "Handling preferences in evolutionary multiobjective optimization: A survey", In: *Proc. of the 2000 Congress on Evolutionary Computation. CEC00 (Cat.No.00TH8512)*, Vol. 1, pp. 30–37, 2000.

[10] D. Bigot, H. Fargier, J. Mengin, and B. Zanuttini, "Probabilistic conditional preference networks", In: *Proc. of 29th Conference on Uncertainty in Artificial Intelligence (UAI 2013)*, 2013.

[11] J. Goldsmith, J. Lang, M. Truszczynski, and N. Wilson, "The computational complexity of dominance and consistency in cp-nets", *J. Artif. Intell. Res. (JAIR)*, Vol. 33, pp. 403–432, 2008.

[12] M. Li, Q. B. Vo, and R. Kowalczyk, "Efficient heuristic approach to dominance testing in cp-nets", In: *Proc. of the 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 1, ser. AAMAS '11. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems*, pp. 353–360, 2011.

[13] E. Alanazi, M. Mouhoub, and S. Zilles, "The complexity of learning acyclic cp-nets", In: *Proc. of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI 2016, New York, NY, USA, pp. 1361–1367, 2016.

[14] E. Alanazi, M. Mouhoub, and S. Zille, "The complexity of exact learning of acyclic conditional preference networks from swap examples", *Artificial Intelligence*, Vol. 278, p. 103182, 2020.

[15] T. E. Allen, J. Goldsmith, H. E. Justice, N. Mattei, and K. Raines, "Uniform random generation and dominance testing for cp-nets", *J. Artif. Int. Res.*, Vol. 59, No. 1, pp. 771–813, 2017.

[16] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit", *IEEE Transactions on Evolutionary Computation*, Vol. 10, No. 5, pp. 477–506, 2006.

[17] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach", *IEEE Transactions on Evolutionary Computation*, Vol. 3, No. 4, pp. 257–271, 1999.

[18] M. Wineberg and F. Oppacher, "Metrics for population comparisons in evolutionary computation systems", *Intelligent systems and control*, 2003.

[19] M. Haqqani and X. Li, "An evolutionary approach for learning conditional preference networks from inconsistent examples", In: *Proc. of International Conference on Advanced Data Mining and Applications. Springer*, pp. 502–515, 2017.

[20] M. Haqqani, H. Ashrafzadeh, X. Li, and X. Yu, "Conditional preference learning for personalized and context-aware journey planning", In: *Proc. of International Conference on Parallel Problem Solving from Nature. Springer*, pp. 451–463, 2018.

[21] S. de Amo, M. L. Bueno, G. Alves, and N. F. F. da Silva, "Mining user contextual preferences", *JIDM*, Vol. 4, No. 1, pp. 37–46, 2013.

[22] A. S. Alashaikh and E. A. Alanazi, "Incorporating ceteris paribus preferences in multiobjective virtual machine placement", *IEEE Access*, Vol. 7, pp. 59 984–59 998, 2019.

[23] X. Sun, Y. Chen, L. Bao, and R. Xu, "Interactive genetic algorithm with implicit uncertainty evaluation for application in personalized search", In: *Proc. of 2017 IEEE Symposium Series on Computational Intelligence*, pp. 1–8, 2017.

[24] J. Branke, "Consideration of partial user preferences in evolutionary multiobjective

optimization", *in Multiobjective optimization. Springer*, pp. 157–178, 2008.

[25] K. Miettinen, *Nonlinear Multiobjective Optimization*, ser. International Series in Operations Research & Management Science. Boston, MA: Springer US, Vol. 12, 1998.

[26] R. Battiti and A. Passerini, "Brain-computer evolutionary multiobjective optimization: A genetic algorithm adapting to the decision maker", *IEEE Transactions on Evolutionary Computation*, Vol. 14, No. 5, pp. 671–687, 2010.

[27] O. M. Shir, M. Preuss, B. Naujoks, and M. Emmerich, "Enhancing decision space diversity in evolutionary multiobjective algorithms", *Evolutionary Multi-Criterion Optimization, M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg*, pp. 95–109, 2009.

[28] M. Preuss, C. Kausch, C. Bouvy, and F. Henrich, "Decision space diversity can be essential for solving multiobjective real-world problems", *Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems, M. Ehrgott, B. Naujoks, T. J. Stewart, and J. Wallenius, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg*, pp. 367–377, 2010.

[29] T. Ulrich, J. Bader, and L. Thiele, "Defining and Optimizing Indicator-Based Diversity Measures in Multiobjective Search", *Parallel Problem Solving from Nature*, PPSN XI. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 707–717, 2010.

[30] T. Ulrich, J. Bader, and E. Zitzler, "Integrating decision space diversity into hypervolume-based multiobjective search", In: *Proc. of the 12th annual conference on Genetic and evolutionary computation - GECCO '10. New York*, New York, USA, p. 455, 2010.

[31] K. Tahernezhadiani, A. Hamzeh, and S. Hashemi, "Towards enhancing solution space diversity in multi-objective optimization: a hypervolume- based approach", *International Journal of Artificial Intelligence & Applications*, Vol. 3, No. 1, p. 65, 2012.

[32] H. Sato, H. Aguirre, and K. Tanaka, "Variable space diversity, crossover and mutation in MOEA solving many-objective knapsack problems", *Annals of Mathematics and Artificial Intelligence*, Vol. 68, No. 4, pp. 197–224, 2013.

[33] J. C. Castillo, C. Segura, A. H. Aguirre, G. Miranda, and C. Leo ́n, "A multi-objective decomposition-based evolutionary algorithm with enhanced variable space diversity control", In: *Proc. of the Genetic and Evolutionary Computation Conference Companion on - GECCO '17. New York*, New York, USA, pp. 1565–1571, 2017.

[34] O. Cuate and O. Schütze, "Variation rate: An alternative to maintain diversity in decision space for multi-objective evolutionary algorithms", *in Evolutionary Multi-Criterion Optimization, K. Deb, E. Goodman, C. A. Coello Coello, K. Klamroth, K. Miettinen, S. Mostaghim, and P. Reed, Eds. Cham: Springer International Publishing*, pp. 203–215, 2019.

[35] D. Mindolin and J. Chomicki, "Hierarchical cp-networks", In: *Proc. of the Third Multidisciplinary Workshop on Advances in Preference Handling (M-PREF)*, Vienna, Austria, 2007.