A Framework for the Study of Preference Incorporation in Multiobjective Evolutionary Algorithms

Raluca Iordache
Dept. of Computer Science
University POLITEHNICA
Bucharest, Romania
riordache@outlook.com

Serban Iordache
SCOOP Software GmbH
Cologne, Germany
siordache@acm.org

Florica Moldoveanu
Dept. of Computer Science
University POLITEHNICA
Bucharest, Romania
florica.moldoveanu@cs.pub.ro

ABSTRACT

We present a formal framework for the study of user preference incorporation into multiobjective evolutionary algorithms. This framework can accommodate virtually any preference model, including those that violate the independence of irrelevant alternatives. We also introduce the Preferanto notation, which permits the specification of a large variety of preference models. A number of properties and indicators are proposed for characterizing preference models. We report the results of a case study experiment assessing the impact of incorporating different preference models into an NSGA-II algorithm.

Categories and Subject Descriptors

Computing methodologies [**Artificial intelligence**]: Search methodologies—*Heuristic function construction*

General Terms

Algorithms

Keywords

preference incorporation, many objective optimization

1. INTRODUCTION

A multiobjective optimization problem (MOP) involves several objectives to be achieved. In most cases, these objectives are conflicting and there is no solution that simultaneously optimizes all of them. Instead, there exists a set of so-called *Pareto-optimal* solutions, which are not dominated by any other feasible solution. Therefore, multiobjective optimizers produce approximation sets consisting of solutions close to the Pareto optimal front. Choosing the best solution from a set of feasible alternatives is not possible without additional preference information, which has to be provided by a decision maker. Depending on the moment when these preferences are articulated, multiobjective optimization methods are based on *a priori*, *a posteriori* or *interactive* approaches [10].

A priori methods involve expressing user preferences before performing the search. Most of these methods are based on aggregating the multiple objectives into a single objective. Typical examples include the *weighted sum* or the *lexicographic* approach. A priori approaches are very appealing, because they transform the multiple optimization problem into a single objective one. However, for real-world complex problems it is in general very difficult or even impossible to aggregate the multiple objectives in a way that accurately reflects the user's preferences.

Interactive methods involve the progressive articulation of preferences. At each generation, the decision makers are asked to provide preference information that will be used to guide the subsequent search. These methods take into account that usually, the decision makers are initially not fully aware of their preferences. However, they gain insight gradually during the solution process, by being exposed to different sets of candidate solutions.

A posteriori methods are concerned with constructing a set of alternative solutions from which the decision makers will choose the one they prefer. The goal of these methods is to provide a set of alternatives that is relatively small, but nonetheless contains all solutions that are most likely of interest to the decision maker. In most cases, this means offering a set of solutions that are well distributed on the Pareto front.

Multiobjective evolutionary algorithms (MOEA) are currently the preferred way to tackle multioptimization problems that are too complex to be solved with traditional operations research techniques. Most research on MOEA focuses on a posteriori methods, and, in particular, on efficient ways of constructing a good set of alternative solutions from which the decision maker will choose the preferred one. In many approaches, the selection pressure is provided only by the Pareto dominance. Such approaches are usually not suitable for many-objective problems, that is, problems involving four or more objectives [11]. When the number of objectives increases, the selection pressure becomes too weak, because the number of non-dominated solutions grows exponentially. The number of solutions needed to approximate the Pareto front also grows exponentially, which makes it very difficult for the decision maker to choose the best alternative.[1]

[1]There are, however, a few approaches able to deal with many-objective problems. See, for example, [2].

GECCO'14, July 12–16, 2014, Vancouver, BC, Canada.

http://dx.doi.org/10.1145/2576768.2598380.

In the recent years, a number of approaches for incorporating user preference information in MOEA have been proposed in order to overcome the above mentioned problems [6, 4, 13]. User preferences refine the partial order induced by the Pareto dominance, focusing the search on the region of interest to the decision maker. Comparing different preference incorporation approaches and deciding which one is the most suitable for a particular problem is a very difficult task. In this paper, we introduce a framework for the study of preference incorporation in MOEA. Our framework can accommodate virtually any preference scheme, including those that violate the independence of irrelevant alternatives. In order to be able to analyze a large variety of preference models, we also introduce the *Preferanto* notation, which is a simple and intuitive language for specifying complex user preferences. A number of properties and indicators are proposed for characterizing preference models. As a proof of concept, we conduct a case study experiment using our framework, in order to assess the impact of incorporating different preference models into an NSGA-II algorithm.

The remaining of this paper is organized as follows. In Section 2 we introduce our preference incorporation framework by formalizing concepts such as *preference model* and *multiobjective problem with preferences*, we describe several properties that may characterize a preference model and we propose a number of preference indicators. In Section 3 we present the Preferanto notation and the methodology of transforming a Preferanto specification into a preference model. In Section 4 we describe the approach used to incorporate preference information into an NSGA-II algorithm, the design of our experiments and their outcome. The last section concludes the paper and outlines future research directions.

2. THE PREFERENCE INCORPORATION FRAMEWORK

In this section, we introduce a formal framework for incorporating preferences into MOEA. Before describing the proposed framework, we briefly review some of the basic concepts and terminology related to multiobjective problems and MOEA.

2.1 Basic concepts

Definition 1. Given a decision space $X \subseteq \mathbb{R}^n$, an objective space $Z \subseteq \mathbb{R}^k$ and a vector of k objective functions $f = (f_1, ..., f_k) : X \to Z$, the *general multiobjective problem* (general MOP) [7] is defined as minimizing $\mathrm{f(x)} = (f_1(\mathrm{x}), ..., f_k(\mathrm{x}))$ subject to:
$g_i(\mathrm{x}) \leq 0,\ i = 1, ..., m$
$h_j(\mathrm{x}) = 0,\ j = 1, ..., p$
where $g_i, h_j : X \to \mathbb{R}$ are the constraint functions of the problem.

Usually, there is no decision vector x that minimizes all the objective functions. Therefore, the solution of a MOP is given by a set of non-dominated decision vectors. A solution $\mathrm{x} \in X$ is said to be *Pareto optimal* with respect to X if $f(\mathrm{x})$ is not dominated by any objective vector $\mathrm{z} \in Z$. The set of all Pareto optimal solutions constitutes the *Pareto optimal set*, while its image in the objective space constitutes the *Pareto optimal front*. These notions can be formalized as follows:

- An objective vector $\mathrm{z} \in Z$ *dominates* an objective vector $\mathrm{z}' \in Z$ (denoted $\mathrm{z} \prec \mathrm{z}'$) iff:
  $z_i \leq z'_i, \forall i \in \{1, ..., k\} \wedge \exists i \in \{1, ..., k\}$ such that $z_i < z'_i$.
- An objective vector $\mathrm{z} \in Z$ *weakly dominates* an objective vector $\mathrm{z}' \in Z$ (denoted $\mathrm{z} \preceq \mathrm{z}'$) iff:
  $z_i \leq z'_i, \forall i \in \{1, ..., k\}$.
- The *Pareto optimal set* $\mathcal{P}^*$ is defined as:
  $\mathcal{P}^* = \{\mathrm{x} \in X \mid \nexists \mathrm{x}' \in X$, such that $f(\mathrm{x}') \prec f(\mathrm{x})\}$
- The *Pareto optimal front* $\mathcal{PF}^*$ is defined as:
  $\mathcal{PF}^* = \{\mathrm{f(x)} \mid \mathrm{x} \in \mathcal{P}^*\}$

2.2 Preference incorporation

In order to give multiobjective evolutionary algorithms (MOEA) the possibility to focus the search on the region of interest, we need to incorporate user preferences in the definition of the MOP. Throughout this paper, we use the term *multiobjective problem with preferences* (MOPP) to denote a MOP enhanced with preference information. Each preference incorporation scheme involves a preference model, for which we propose the following definition:

Definition 2. A function $\mathscr{P} : 2^Z \to 2^{Z \times Z}$ is called a *preference model* over the objective space Z, iff $\forall A \subseteq Z$, $\mathscr{P}(A)$ is a strict partial order on A.

Given a set $A \subseteq Z$ and two objective vectors $\mathrm{u}, \mathrm{v} \in A$, we say that the vector u *preferentially dominates* the vector v with respect to A (denoted $\mathrm{u} \underset{A}{\prec\cdot} \mathrm{v}$), if $(\mathrm{u}, \mathrm{v}) \in \mathscr{P}(A)$.

Definition 3. Given a preference model $\mathscr{P}$ over an objective space Z and a set $A \subseteq Z$, the *preferential front* of A (denoted $\mathcal{F}^*_A$) is defined as the set of preferentially non-dominated vectors with respect to A:
$$\mathcal{F}^*_A = \{\mathrm{u} \in A \mid \nexists \mathrm{v} \in A, \text{ such that } \mathrm{v} \underset{A}{\prec\cdot} \mathrm{u}\}$$

A preference model can be attached to a MOP only if it does not violate the Pareto dominance relation imposed by its objective functions.

Definition 4. A preference model $\mathscr{P}$ is *Pareto compliant* with a given MOP iff
$\forall A \subseteq Z, \forall \mathrm{u}, \mathrm{v} \in A, \mathrm{u} \underset{A}{\prec\cdot} \mathrm{v} \Rightarrow \mathrm{v} \not\prec \mathrm{u}$.

We propose the following definition, which describes a multiobjective problem with preferences (MOPP) as a MOP together with a Pareto compliant preference model.

Definition 5. Given a decision space $X \subseteq \mathbb{R}^n$, an objective space $Z \subseteq \mathbb{R}^k$, a vector of k objective functions $f = (f_1, ..., f_k) : X \to Z$ and a Pareto compliant preference model $\mathscr{P}$ over the objective space Z, the *general multiobjective problem with preferences* (general MOPP) is defined as minimizing $\mathrm{f(x)} = (f_1(\mathrm{x}), ..., f_k(\mathrm{x}))$, with $f(x) \in \mathcal{F}^*_Z$, subject to:
$g_i(\mathrm{x}) \leq 0,\ i = 1, ..., m$
$h_j(\mathrm{x}) = 0,\ j = 1, ..., p$
where $\mathcal{F}^*_Z$ is the preferential front of the objective space Z and $g_i, h_j : X \to \mathbb{R}$ are the constraint functions of the problem.

It can be noticed that the above definition could be rewritten without referring to a preferential model, because $\mathcal{F}_Z^*$ only involves the preferential dominance relation with respect to Z. This means that a partial order on Z would be sufficient to define the MOPP. The reason for formulating the definition in terms of a preferential model is our interest in evolutionary algorithms for the MOPP. Such algorithms deal with populations of candidate solutions at each generation step. A preference model is useful in evaluating the fitness of the individuals in a population. It also helps assessing the performance of different evolutionary algorithms for the MOPP.

- A preference model $\mathscr{P}$ is *Pareto complete* with respect to a given MOP iff:
  $\forall A \subseteq Z, \forall \mathrm{u}, \mathrm{v} \in A, \mathrm{u} \prec \mathrm{v} \Rightarrow \mathrm{u} \underset{A}{\prec\cdot} \mathrm{v}$.
- A preference model $\mathscr{P}$ is called *total* with respect to a given MOP iff:
  $\forall A \subseteq Z, \mathscr{P}(A)$ is a strict total order on A.
- A preference model $\mathscr{P}$ is *weakly consistent* with respect to a given MOP iff:
  $\forall A, B \subseteq Z, \forall \mathrm{u}, \mathrm{v} \in A \cap B, \mathrm{u} \underset{A}{\prec\cdot} \mathrm{v} \Rightarrow \mathrm{v} \underset{B}{\not\prec\cdot} \mathrm{u}$.
- A preference model $\mathscr{P}$ is *strongly consistent* with respect to a given MOP iff:
  $\forall A, B \subseteq Z, \forall \mathrm{u}, \mathrm{v} \in A \cap B, \mathrm{u} \underset{A}{\prec\cdot} \mathrm{v} \Rightarrow \mathrm{u} \underset{B}{\prec\cdot} \mathrm{v}$.

Non-consistent preferences violate the *independence of irrelevant alternatives* (IIA). The concept of IIA has been introduced by Arrow [1], and is one of the criteria considered in his *Impossibility theorem*. The violation of the IIA is also known as the *rank reversal* phenomenon. We can illustrate the problem addressed by the IIA considering the following scenario:

Faced with two alternatives A and B, a person prefers A to B. After introducing a third alternative C, the person prefers B to A.

The person in the above scenario expresses preferences that do not satisfy the IIA criterion.

To our best knowledge, non-consistent preferences have not been studied in the context of MOEA, although there are several multicriteria decision making (MCDM) approaches affected by rank reversal (see [18] for an illustration of the rank reversal phenomenon in 5 popular MCDM approaches). The legitimacy of rank reversal in MCDM is a controversial issue [14] and, obviously, having to deal with rank reversal is not desirable. However, in many real world situations, it may be impossible to avoid this phenomenon. Experimental evidence indicates that the IIA is often violated by decision makers [16]. Therefore, it is important that our framework also accommodates non-consistent preferences.

2.3 Preference indicators

In the previous subsection we have defined a number of possible traits of a preference model (Pareto completeness, totalness, consistency). A more accurate characterization can be obtained by defining scalar measures associated with a preference model.

Definition 6. A *preference indicator* is a function
$I_{\mathrm{pref}} : \mathscr{P} \to \mathbb{R}$
that associates numerical values to preference models.

The preference indicators we propose in this subsection are based on the properties of the partially ordered sets (posets) obtained by applying a preference model to a given set. Therefore, it is helpful to have corresponding scalar measures associated with posets.

Definition 7. A *poset indicator* is a function
$I_{poset} : 2^{Z \times Z} \to \mathbb{R}$
that associates numerical values to strict partial orders:
$\forall R \subseteq Z \times Z$, such that R is a strict partial order,
$I_{\mathrm{poset}}(R) \in \mathbb{R}$ is the *poset indicator* of R.

In the context of our preference incorporation framework, the strict partial orders R referred in the above definition are obtained by applying a preference model to an approximation set A. Therefore, if $u, v \in A$ and uRv, we can say that solution u preferentially dominates solution v.

2.3.1 Selectivity indicators

We are interested in assessing the degree of refinement introduced by a preference model, by analyzing the strict posets which it produces. We will refer this property of a preference model as its *selectivity* and we introduce the following 3 *poset selectivity indicators*:

1. **The dominance poset indicator**
   Defined as the number of solutions that are preferentially dominated by at least one other element, divided by the number of solutions.
   $\forall A \subseteq Z, \forall R \subseteq A \times A,$
   $I_{\mathrm{poset}}^{\mathrm{dominance}}(R) = |\{u \in A \mid \exists v \in A, vRu\}| \ / \ |A|$

2. **The relation poset indicator**
   Defined as the number of preferentially domination relations divided by the maximum number of possible preferentially domination relations.
   $\forall A \subseteq Z, \forall R \subseteq A \times A,$
   $I_{\mathrm{poset}}^{\mathrm{relation}}(R) = |\{(u, v) \in A \times A \mid uRv\}| \ / \ M,$
   where $M = \frac{|A| * (|A|-1)}{2}$.

3. **The front poset indicator**
   Defined as the number of fronts generated by the non-dominating sorting procedure used by the NSGA algorithms [15] minus 1, divided by the number of solutions.
   $\forall A \subseteq Z, \forall R \subseteq A \times A,$
   $I_{\mathrm{poset}}^{\mathrm{front}}(R) = (\text{number-of-fronts} - 1) \ / \ |A|.$

Intuitively, we can define the *selectivity* I_{pref}^{sel} of a preference model $\mathscr{P}$ with respect to a poset selectivity indicator I_{poset}^{sel} as the average poset selectivity value over the set of all non-dominated subsets of Z. However, such a definition is not mathematically rigorous and, in the general case, it is also not a reliable measure. We define instead a procedure for estimating the selectivity of a preference model, which is appropriate for our preference incorporation framework.

The selectivity of a preference model is considered a reliable indicator only if the poset selectivity values considered for computing its value have low variance. For the three poset selectivity indicators introduced above, our experiments show that their values are affected by the size of the approximation sets considered. However, these values have low variance for approximation sets of the same

size. Therefore, the estimation procedure presented in Algorithm 1 provides a reliable measure for the selectivity of a preference model, for a given approximation set size. Choosing the size to be used in estimating the selectivity of a preference model is usually straightforward, because most MOEA use a fixed population size during their evolution. If this is not the case, a value close to the maximum population size should be chosen.

Algorithm 1 Estimate the selectivity of a preference model

procedure COMPUTESELECTIVITY($\mathscr{P}, I^{\text{sel}}_{\text{poset}}, N, S$)
  generate N approximations of the Pareto front,
    each set having the cardinality S
  $sum \leftarrow 0$
  for each approximation set A **do**
    $sum \leftarrow sum + I^{\text{sel}}_{\text{poset}}(\mathscr{P}(A))$
  end for
  $selectivity \leftarrow sum/N$
end procedure

2.3.2 Inconsistency indicators

In many cases, non-consistent preferences are unavoidable, although they lead to the undesirable rank reversal phenomenon. This means that the preference for a solution over another is context dependent, that is, it is affected by the other solutions present in the candidate set. We consider that a few occurrences of this phenomenon are acceptable, but we discourage the incorporation of preference models with a high rate of rank reversals, because they have a rather chaotic character.

In order to measure the degree of inconsistency exhibited by a preference model, we introduce a class of preference indicators called *inconsistency indicators*. As in the case of selectivity indicators, we offer a procedure for estimating the value of an inconsistency indicator for a given approximation set size. This procedure makes use of a *discrepancy operator* $\delta : 2^{Z\times Z} \times 2^{Z\times Z} \rightarrow [0,1]$. For two strict posets R and Q on a set A, $\delta(R,Q)$ gives the discrepancy between these posets.

Various discrepancy operators can be defined. A simple one, based on the number of differences between the compared posets is presented below:

$\delta_{\text{diff}}(R,Q) = |\{(u,v) \in A \times A \mid uRv \neq uQv\}| \ / \ (|A| * (|A|-1))$

Given an approximation set A and a discrepancy operator δ, the poset inconsistency indicator is defined as:

$I^{\text{inconsist}}_{\text{poset}}(A,\delta) = (\sum_{z\in A} \delta(\mathscr{P}(A) \restriction_{(A\setminus\{z\})}, \mathscr{P}(A\setminus\{z\}))) \ / \ |A|$

The symbol $\restriction$ in the above equation denotes the restriction of a binary relation: given a poset R on a set A and a subset $B \subset A$, $R \restriction_B$ is the restriction of R to the subset B.

The Algorithm 1 can be reused to estimate the inconsistency indicator of a preference model, by replacing the $I^{\text{sel}}_{\text{poset}}$ with the $I^{\text{inconsist}}_{\text{poset}}$.

3. THE PREFERANTO NOTATION

Current approaches of incorporating decision maker preferences into MOEA are constrained to a particular formulation of preferences [12]. The method of dealing with the particular preference scheme is perceived as part of the search methodology, although, in many cases, the search algorithm is independent of the preference scheme used and could be adapted straightforwardly to work with any preference model.

The theoretical framework introduced in the previous section treats the preference model as a parameter of the optimization problem. This view fosters an approach that clearly separates the search-related aspects from those concerning the handling of some particular preference scheme, thus allowing algorithm designers to concentrate on the core of their work. This is in line with recent efforts to separate the algorithm-specific part of an optimizer from the application-specific part (see, for example, the PISA framework [3]).

Since the preference model is treated as a parameter of the optimization problem, it would be useful to have a standard way to specify it. The benefit of a unified way of expressing preferences has been recognized by Purshouse et al. [12] and a first step in this direction has been made by Wang et al. [17]. In this section, we propose a specification language called *Preferanto*, which allows describing a large variety of preference models using a simple but powerful notation. Preferanto can be easily integrated into existing multiobjective test problem toolkits in order to provide multiobjective problems with preferences. Therefore, Preferanto is a useful companion tool to our theoretical framework, which allows experimenting with various preference models and incorporation schemes.

3.1 Notation

We introduce the Preferanto notation gradually, starting with a preference model using the lexicographic approach. In this approach, the objectives are ranked by their importance. When comparing two solutions sol_1 and sol_2, their objectives are compared in the order of their ranks. If, for the first objective at which their values differ, sol_1 is better than sol_2, then sol_1 preferentially dominates sol_2 ($sol_1 \prec\cdot\ sol_2$).

Let us consider a MOP with the objectives z_1, z_2 and z_3, which are ranked in the order: z_3, z_1, z_2. Preferanto is a rule based specification language, which requires each rule to be written on a separate line, in the order of their priorities. Our simple preference model will therefore consist of 3 rules, expressed in the Preferanto notation as follows:

```
preferences {
  z3;
  z1;
  z2;
}
```

A Preferanto rule may involve several objectives, combined in a mathematical expression. This way, a weighted sum preference model can be expressed using a single rule, containing the aggregation formula. A weighted sum preferential model with the weights 0.1, 0.2 and 0.3 can be represented as:

```
preferences {
  0.1 · z1 + 0.2 · z2 + 0.3 · z3;
}
```

Moreover, a Preferanto rule is not restricted to a single mathematical expression. Instead, it can contain a tuple of such expressions, as in the following specification:

```
preferences {
  < z1 + 0.5 · z2, z3 >;
  < z2, z3 + 0.7 · z1, z1 · z2 >;
}
```

In the above example, the comparison of two solutions sol_1 and sol_2 starts with the first rule, which implies computing

the values of $z_1+0.5\cdot z_2$ and z_3 for both solutions. If, for both expressions the values of sol_1 are better than the ones of sol_2, then sol_1 preferentially dominates sol_2. Conversely, if for both expressions the values of sol_2 are better than the ones of sol_1, then sol_2 preferentially dominates sol_1. Otherwise, the second rule is considered in an analogous way.

User preferences such as those considered in the Guided Multi-Objective Evolutionary Algorithm (G-MOEA) [5] can be easily expressed in Preferanto by using a single rule specification with a tuple with two elements:

```
preferences {
  < z1 + a12 · z2, a21 · z1 + z2 >;
}
```

A Preferanto rule having a single mathematical expression can be seen as a rule with a tuple that contains only one element. Therefore, all Preferanto rules involve tuples. The angle brackets are optional for tuples with only one element.

A Preferanto rule can also have attached a condition. The effect of the condition is that the rule is taken into consideration only if the condition evaluates to *true*. This allows expressing the fact that some rules become important only in a given context. Let us consider a hypothetical e-commerce company confronted with a multiobjective problem involving shipping goods to its customers. Two objectives are relevant for the company in choosing the best shipping alternative for a product: the shipping cost and the delivery time. The company sees the shipping cost as the most important objective. However, the company has to pay a substantial penalty if the good is not delivered within 7 days. Therefore, the delivery time becomes the most important objective, if one of the solutions compared has a delivery time greater than 7 days.

In order to express conditions as the one needed in the above scenario, Preferanto makes use of three *preferential operators*, as seen in Table 1. The preferential operators

Table 1: Preference operators

Preference operator	Meaning
AT_LEAST_ONE(cond)	cond(sol_1) OR cond(sol_2)
EXACTLY_ONE(cond)	cond(sol_1) XOR cond(sol_2)
ALL(cond)	cond(sol_1) AND cond(sol_2)

take as argument a boolean formula, which usually involves one or many objectives. The formula is evaluated twice, once for each of the solutions to be compared. The two resulting boolean values are passed as arguments to the boolean operator (OR, XOR, or AND) associated with the given preference operator, in order to decide whether the corresponding rule should be taken into consideration or skipped.

In the e-commerce company scenario discussed above, the delivery time becomes the most important objective only if exactly one of the two solutions compared has a delivery time greater than 7 days. If both solutions have lower delivery times, there is no penalty. If both have greater delivery times, the penalty must be paid no matter which solution is chosen. Therefore, the preferences for this scenario can be expressed using the EXACTLY_ONE operator as shown below:

```
preferences {
  [EXACTLY_ONE(deliveryTime > 7)] deliveryTime;
  cost;
  deliveryTime;
}
```

As seen above, the condition attached to a rule is written in brackets. It is allowed to construct conditions that combine several preferential operators, as in the following example involving the quality of service of a network:

```
preferences {
  [AT_LEAST_ONE(availability < 0.98) &
            ALL(cost < 5)] availability:high;
  <cost, responseTime>;
}
```

For convenience, Preferanto also defines two *direction operators*: *low* and *high*. In the example above, the *high* operator is used to indicate that *availability* is a maximization objective. The *low* operator is default and can be omitted, but using it to specify preferences that combine minimization and maximization objectives improves readability.

3.2 Transforming a Preferanto specification into a preference model

Our goal in this subsection is to devise a general procedure for converting a Preferanto specification into a preference model. Taking into account the definition of a preference model, this goal can be reformulated as devising a general procedure such that: for any Preferanto specification and any population A, the procedure constructs a strict partial order on A, which is in accordance with the Preferanto specification.

We start by describing an algorithm for pairwise comparisons based on a Preferanto specification (Algorithm 2).

Algorithm 2 Pairwise comparison of two solutions

```
function PAIRWISECOMPARE(sol1, sol2, preferences)
  for each rule r ∈ preferences do
    if cond_r is null OR cond_r(sol1, sol2) = true then
      result ← compare(tuple_r(sol1), tuple_r(sol2))
      if result ≠ 0 then return result
      end if
    end if
  end for
  return 0
end function
```

The pairwise comparison is implemented as a function returning -1 if sol_1 is preferred to sol_2, 1 if sol_2 is preferred to sol_1 and 0 if there is no preference relation between the two solutions. The `pairwiseCompare` function processes the rules in the order of their priorities. If the current rule has no condition or its condition evaluates to true, the tuples corresponding to sol_1 and sol_2 for this rule are compared using the function `compare`. This function returns -1 if all elements in the tuple of sol_1 have better values than the elements in the tuple of sol_2, 1 if all elements in the tuple of sol_2 have better values than the elements in the tuple of sol_1, and 0 otherwise. If `compare` returns a non-zero value, this value is used as the return value of `pairwiseCompare`. If the tuple comparisons performed by `compare` return 0 for all rules, `pairwiseCompare` returns 0.

The pairwise comparisons performed by Algorithm 2 impose a binary relation on the set of individuals in a population A. In what follows, such a relation will be referred as the *pairwise comparison relation on* A, denoted PCR_A. It can be easily proved that in the case of a Preferanto specification having only unconditional rules, PCR_A is a strict partial order, $\forall A$. Therefore, a Preferanto specification hav-

ing only unconditional rules can be transformed into a preference model by using its pairwise comparison relations as follows: $\forall A, \mathscr{P}(A) = PCR_A$.

For a Preferanto specification having at least one conditional rule, PCR_A is in general not transitive and therefore it is not a strict partial order. We exemplify this by considering the following Preferanto specification:

```
preferences {
  [ALL(z1 < 0.5)] z1;
  z2;
}
```

We consider a set A containing three solutions (sol_1, sol_2, sol_3), with the objective values given in Table 2.

Table 2: Objective values for a set of solutions A leading to an intransitive PCR_A

Objective	sol_1	sol_2	sol_3
z_1	0.2	0.4	0.6
z_2	0.7	0.3	0.5

The PCR for these values and the above Preferanto specification includes the following intransitive preferences: $sol_1 \prec\cdot\ sol_2$, $sol_2 \prec\cdot\ sol_3$, $sol_3 \prec\cdot\ sol_1$.

Since PCR is not transitive in general, another approach has to be taken in order to transform a Preferanto specification having conditional rules into a preference model. This approach will be discussed in the remaining of this subsection.

For any Preferanto specification having conditional rules and any approximation set A, we need to construct a strict partial order on A. If A has n solutions, any relation on A can be represented by a matrix R with $n \times n$ elements. Therefore, our goal is to devise a function `getPoset`, which takes as arguments an approximation set A and a preference specification, and returns a matrix R representing the strict partial order induced on A. The matrix R used in our approach stores not only information about the preference relation between solutions, but also about the preference rule that has been decisive in establishing this relationship:

$R_{ij} = r$, iff $sol_i \prec\cdot\ sol_j$ due to the rule r.
$R_{ij} = -r$, iff $sol_j \prec\cdot\ sol_i$ due to the rule r.

In order to ensure that at each step the matrix R corresponds to a strict partial order, all changes to R will be made through the function `setTrans` presented in Algorithm 3.

The function `setTrans` returns $false$ if by adding the preference relation given as argument and taking the transitive closure, R is no longer a partial order (for example, because the newly added preference relation has produced a cycle).

The basic idea for implementing the `getPoset` function is to iterate through the preference rules and at each step to create a list with the candidate preference relations induced by the current rule. Then, the preference relations from this list will be added sequentially to R using the `setTrans` function, until all preference relations have been added or a preference relation that would break the strict partial order has been found. After founding a preference rule that breaks the partial order, this preference relation and all the remaining ones in the list will be skipped and the next preference rule will be considered.

If used in the above form, the outcome of `getPoset` would depend on the order in which the preference relations have been added to the current list, that is, on the order in which the pairwise comparisons have been performed. Since this is not admissible, we need a method of sorting the list of candidate preference relations. To this end, we store additional information in the list of candidate preference relations, in the form of an array of differences between the expressions defining each element in the tuple of the current rule. Specifically, let us consider that the current rule has a tuple $< expr_1, expr_2, ..., expr_m >$. During the pair comparison of two solutions sol_i and sol_j, we create an m-dimensional array `diffs`, such that:

$\texttt{diffs}[k] = expr_k(sol_i) - expr_k(sol_j), \forall k = 1..m$ The elements of the list of candidate preference relations will be data structures of the type `PrefRel`, which contains the following fields: (`from`, `to`, `diffs`). The field `from` gives the index of the dominating solution, the field `to` gives the index of the dominated solution and the field `diffs` is the array mentioned before. The list of candidate preference relations can now be sorted lexicographically based on the values in the `diffs` arrays.

The pseudocode of the `getPoset` function is presented in Algorithm 4 and the pseudocode of the `getCandidatePreferenceRelations` function is presented in Algorithm 5.

In general, the preference models constructed using the function `getPoset` are not consistent, that is, they are affected by the rank reversal phenomenon. Preferanto is thus a powerful notation, capable to express both consistent preference models (by using only unconditional rules) and preference models that violate the independence of irrelevant alternatives (by including at least one conditional rule).

Algorithm 3 Adding a preference relation to R and ensuring that R remains a strict partial order

```
function setTrans(R, i, j, r)
  if R_ij ≠ 0 then
    if R_ij = r then
      return true
    else
      return false
    end if
  end if
  R_ij ← r
  R_ji ← −r
  R ← transitive closure of R
  if R is not a strict partial order then
    return false
  else
    return true
  end if
end function
```

4. CASE STUDY

In order to illustrate the usefulness of our framework, we devise an experiment for assessing the impact of incorporating different preference models into an NSGA-II algorithm [8]. NSGA-II uses a nondominated sorting algorithm in order to construct a series of fronts. Each individual in the front i is dominated by all individuals in any front $p < i$ and dominates all individuals in any front $q > i$. No domination relation exists between individuals in the same front. The fitness of an individual is given by the front on which it re-

Algorithm 4 Obtaining a strict poset for A in accordance with the Preferanto $preferences$

```
function getPoset(A, preferences)
    initialize R
    for each rule r ∈ preferences do
        list ← getCandidatePreferenceRelations(R, r)
        sort list lexicographically based on the diffs field
        Q ← R
        lastDiffs ← null
        for each prefRel ∈ list do
            if lastDiffs ≠ null AND lastDiffs ≠ diffs then
                R ← Q
            end if
            ok ← setTrans(Q, prefRel_from, prefRel_to, r)
            if ok = false then
                Skip the remaining elements in list
                Continue with the next rule
            end if
        end for
    end for
    return R
end function
```

Algorithm 5 Creating the list of candidate preference relations

```
function getCandidatePreferenceRelations(R, r)
    initialize list
    for i ← 1, n − 1 do
        for j ← i + 1, n do
            diffs ← null
            if cond_r is null OR cond_r(sol_i, sol_j)=true then
                diffs ← tuple_r(sol_i) − tuple_r(sol_j)
                if all elements of diffs are negative then
                    list ← list + PrefRel(i, j, -diffs))
                end if
                if all elements of diffs are positive then
                    list ← list + PrefRel(j, i, diffs))
                end if
            end if
        end for
    end for
    return R
end function
```

sides. The NSGA-II algorithm uses the Pareto dominance in order to construct the fitness fronts. In order to incorporate user preferences, we replace the Pareto dominance with a dominance relation that chains Pareto dominance and preferential dominance:
$\forall A, \forall u, v \in A, u$ dominates v iff $u \prec v$ or $(v \not\prec u$ and $u \underset{A}{\prec} v)$.

In what follows, we will use the acronym NSGAP (NSGA with preferences) to refer to our NSGA-II algorithm enhanced with preference information. In order to implement the NSGAP algorithm, we have used the MOEA Framework (`http://www.moeaframework.org/`). The source code of our implementation is freely available at `https://github.com/preferanto/preferanto`.

Our goal is to study how preference models with different selectivities affect the performance of the NSGAP relative to the NSGA-II. In all experiments we use the *dominance selectivity indicator* and a population size of 100. The problem considered is DTLZ2 [9], which is scalable to any number of decision variables and to any number of objectives. For both algorithms we have considered all DTLZ2 problems with the number of objectives between 2 and 10 and we have used as termination condition a number of 5000 function evaluations. The number of decision variables has been set as 9 + the number of objectives.

Our first step is to produce for each number of objectives sets of preference models with the following selectivities: 0.1, 0.2, ..., 0.9. By using a simple genetic algorithm that generates preference models and evaluates their selectivity, we have produced for each selectivity in the considered range a set of 10 different preferential models. The preference models used as individuals by the genetic algorithm have been obtained by generating Preferanto notations with only unconditional rules. Each rule is a tuple with a variable number of elements. Each element in a tuple is a weighted sum of a variable number of objectives. The genetic algorithm has therefore to solve a single objective optimization problem, where the objective is given by the desired selectivity and the decision variables are the parameters of the Preferanto notations: the number of rules, the number of elements in a tuple, the number and weights of the terms in an element of a tuple.

In a second step, we have generated and stored reference sets for both NSGA-II and NSGAP. For each number of objectives we have produced 10 approximation sets using the NSGA-II algorithm. For each number of objectives, for each selectivity and for each of their 10 associated preference models, we have generated a reference set using the NSGAP algorithm.

In the final step we have computed for each objective count and each selectivity a *nondominance ratio indicator*, using the following procedure: For a given preference model and a given reference set produced by the NSGA-II algorithm, we preferentially compare each solution in the NSGA-II approximation set with each solution in the NSGAP approximation set and we count how many of the solutions in the NSGA-II front are not dominated by any solution in the NSGAP approximation front. This value is then divided by the number of solutions in the NSGA-II front in order to obtain the *nondominance ratio indicator*. Finally, for each number of objectives and each selectivity we average the nondominance ratio indicators computed for their corresponding 10 different preference models.

In the absence of preference information, which corresponds to a selectivity with value 0, all solutions are preferentially nondominated, and the nondominance ratio indicator has the value 1.

Figure 1 shows how the nondominance ratio indicator is affected by the selectivity for 2, 3 and 10 objectives. It can be observed that the impact of incorporating user preferences becomes more significant when the number of objectives increases. For 10 objectives, even a small value of selectivity such as 0.1 leads to an abrupt fall of the nondominance ratio. This means that even when preference information is scarce, the incorporation of this information in MOEA may lead to a dramatic performance boost.

5. CONCLUSIONS

We have introduced a framework for the study of user preference incorporation in MOEA and a preference speci-

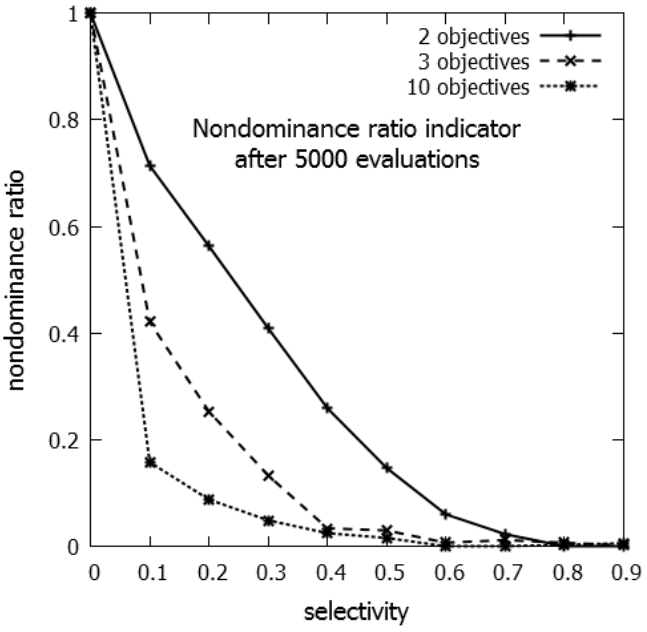

Figure 1: Nondominance ratio indicator after 5000 function evaluations.

fication notation that allows expressing complex preference models. The case study shows that the selectivity indicators defined by our framework are reliable and useful measures for characterizing preference models.

We have devised a method of transforming a Preferanto specification into a preference model. For specifications containing only unconditional rules, the method produces a consistent preference model, while for specifications that contain at least one conditional rule, the resulting preference model may be affected by the rank reversal phenomenon.

Incorporating non-consistent preferences and assessing the performance of the algorithms using such preferences are tasks that pose a series of challenges, due to the rank reversal problem. Further research has to be done in order to analyze different characteristics of non-consistent preference models, such as the impact of set size and selectivity on the rate of rank reversals and how these rank reversals are distributed among the most preferred and least preferred solutions.

Future research directions also include: devising more sophisticated preference indicators; developing faster methods of estimating the value of a preference indicator (for example, based on the Preferanto specification of a preference model); finding methods to determine whether a Preferanto specification with conditional rules is affected by the rank reversal phenomenon.

6. REFERENCES

[1] K. J. Arrow. *Social Choice and Individual Values*. John Wiley and Sons, New York, NY, 1951.

[2] J. Bader and E. Zitzler. HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. *Evolutionary Computation*, 19(1):45–76, Spring, 2011.

[3] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA—A Platform and Programming Language Independent Interface for Search Algorithms. In *EMO 2003*, pages 494–508, Faro, Portugal, 2003. Springer. LNCS. Volume 2632.

[4] J. Branke and K. Deb. Integrating user preferences into evolutionary multi-objective optimization. In Y. Jin, editor, *Knowledge Incorporation in Evolutionary Computation*, pages 461–478. Springer, October 2004.

[5] J. Branke, T. Kaußler, and H. Schmeck. Guidance in evolutionary multi-objective optimization. *Advances in Engineering Software*, 32:499–507, 2001.

[6] C. A. C. Coello. Handling preferences in evolutionary multiobjective optimization: a survey. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 1, 2000.

[7] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

[9] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable Test Problems for Evolutionary Multi-Objective Optimization. Technical report, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH Zurich), 2001.

[10] J. Horn. *Handbook of Evolutionary Computation*, chapter Multicriterion decision making. IOP Press, Oxford, 1997.

[11] R. Purshouse and P. Fleming. Evolutionary many-objective optimisation: An exploratory analysis. In *IEEE Congress on Evolutionary Computation, CEC'03*, volume 3, pages 2066–2073, 2003.

[12] R. C. Purshouse, K. Deb, M. M. Mansor, S. Mostaghim, and R. Wang. A review of hybrid evolutionary multiple criteria decision making methods. *COIN Report*, (2014005), January 2014.

[13] L. Rachmawati and D. Srinivasan. Preference incorporation in multi-objective evolutionary algorithms: A survey. In *IEEE Congress on Evolutionary Computation (CEC 2006)*, pages 962–968, Vancouver, BC, 2006.

[14] T. L. Saaty and L. G. Vargas. The legitimacy of rank reversal. *Omega*, 12(5):513–516, 1984.

[15] N. Srinivas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2(3):221–248, 1994.

[16] A. Tversky and I. Simonson. Context-Dependent preferences. *Manage Sci Manage Sci*, 39(10):1179–1189, 1993.

[17] R. Wang, R. C. Purshouse, and P. J. Fleming. "Whatever Works Best for You"- A New Method for a Priori and Progressive Multi-objective Optimisation. In *EMO 2013*, pages 337–351. Springer. LNCS Vol. 7811, Sheffield, UK, 2013.

[18] Y.-M. Wang and Y. Luo. On rank reversal in decision analysis. *Mathematical and Computer Modelling*, 49(5-6):1221–1229, 2009.