Interactive Pareto local search with imprecise trade-offs

Thibaut Lust[1]

Abstract. In this paper, we study the integration of imprecise trade-offs into multi-objective combinatorial optimization. We focus our study on many-objective optimization, where at least three objectives have to be optimized. Imprecise trade-offs are used to reduce the size of the Pareto set by considering preferences expressed by the decision maker. The new dominance relation is learned through an interactive method, based on the Pareto local search heuristic. Every time that the number of non-dominated solutions produced by the Pareto local search is higher than an initial number, we ask the decision maker to compare two alternatives to refine the preference relation. We intensively study this method on the multi-objective traveling salesman problem. Results are given for instances with up to 200 cities and 6 objectives and we show that better results can be obtained than with a Pareto approach, with significant speed-up factors.

1 Introduction

When solving a multi-objective optimization problem, the main goal is to help a decision maker (DM) to find a solution or a small set of solutions that correspond to his/her preferences. These solutions are Pareto optimal solutions (none of the objective values can be improved without degrading at least one of the others), for which the DM is convinced that they correspond to the best options possible. Of course, this implies the participation of the DM who should give some insights into the problem and shares his/her preferences.

In this work, we focus our study on many-objective combinatorial optimization [7], where at least three objectives have to be optimized. When considering many-objective combinatorial optimization problems, only using Pareto dominance leads to huge sets of Pareto efficient solutions, often intractable [15]. It is thus important to refine the Pareto dominance by integrating preferences of the DM. Note that another solution, not considered in this paper, is to restrict the size of the Pareto set for obtaining a well-represented Pareto set. This is often based on a division of the objective space into different regions (and only one solution represents a region) [19] or on ϵ-dominance [21].

We will here refine the Pareto dominance relation with preferences and more specifically with imprecise trade-offs. Indeed, in many situations, the DM allows some trade-offs between the objectives. For example, maximally acceptable trade-offs of the form "one unit improvement of objective i is worth at most a_{ji} degradation units in objective j" are often possible.

Using imprecise trade-offs in multi-objective optimization has been introduced by Branke and Deb [6] in the context of evolutionary multi-objective optimization. They integrate imprecise trade-offs into the evolutionary algorithm NSGA-II [13] by modifying the crowding distance of the original algorithm, but only continuous problems with two objectives are considered. Another group of authors [26, 27, 34] have also integrated imprecise trade-offs, but in the context of multi-objective influence diagrams and multi-objective constraint optimization. In these works, the imprecise trade-offs are known before running the algorithm and they develop exact methods to find the solutions that are optimal regarding the *a priori* preference relation.

In this paper, we consider an interactive method where imprecise trade-offs are learned during the exploration of the solutions. Interactive methods are now regarded as the most promising methods to solve multi-objective problems [28, 29] due to numerous advantages: they are generally fast, the learning process is integrated in the optimization process and one or a few number of solutions are produced, corresponding to the preferences of the DM. The DM does not need to have any global preference structure before running the algorithm. However, in our approach (s)he will have to decide the maximal number of Pareto optimal solutions to generate. Indeed, we do not limit the method to the search of only one preferred solution since in some cases robustness considerations have to be taken into account when there exist some uncertainties, imprecision or inconsistencies in the data or in the model [28].

We will integrate imprecise trade-offs in a popular multi-objective heuristic: Pareto local search (PLS) [30]. Two reasons justify this choice: first, PLS is used as a crucial component in some of the best heuristics for solving multi-objective combinatorial optimization problems: state-of-the-art results obtained for knapsack [24], traveling salesman [11] and set covering [25] problems all used PLS. The general idea of such methods is to start PLS from a set of high quality solutions generated by some other methods, e.g. the powerful Lin-Kernighan heuristic for TSP [22]. PLS has also been used in other contexts, e.g. for solving scheduling problems [10], multi-agent problems [17] or multi-objective Markov decision processes [20]. Moreover, PLS uses directly the Pareto dominance relation to search for new non-dominated solutions. Therefore, a simple adaptation of PLS to take into account imprecise trade-offs will be to replace the Pareto dominance relation by the refined dominance relation based on the trade-offs introduced by the DM. We will see, however, that further adaptations are needed to attain high-quality results.

The contributions of the paper are three-fold: we first show how to learn imprecise trade-offs by asking questions to the decision maker. We then propose an interactive version of PLS with a dominance relation based on trade-offs allowing to solve high size instances with high number of objectives, in low computational times. We finally present results for many-objective traveling salesman problem instances, with up to 6 objectives. We compare our results with a Pareto approach and we show that superior results are obtained.

The paper is organized as follows: we first define multi-objective optimization problems and present some general methods for dealing with such problems (Section 2). We then introduce imprecise trade-offs and how to compute easily dominance calculations with trade-offs

[1] CNRS, Laboratoire d'informatique de Paris 6, LIP6, Sorbonne Université, 75005 Paris, France, email: thibaut.lust@lip6.fr

(Section 3). An adaptation of PLS is then proposed (Section 4) and intensive experiments are conducted on the many-objective traveling salesman problem in Section 5.

2 Multi-objective optimization

We consider in this paper a general multi-objective optimization problem with m objective functions to minimise, defined as follows: "$\underset{x\in\mathcal{X}}{\text{minimise}}$"$\big(y_1(x), y_2(x), \ldots, y_m(x)\big)$. In this definition, $\mathcal{X}$ is the feasible set in the decision space, typically defined by some constraint functions. The image of the feasible set in objective space is called $\mathcal{Y} \subset \mathbb{R}^m$ and is defined by $\mathcal{Y} = y(\mathcal{X})$ where $y(x) = (y_1(x), y_2(x), \ldots, y_m(x))$, that is a solution is evaluated according to the m objective functions. Solutions are usually compared according to *Pareto dominance* through their images in the objective space (called points): we say that a point $u = (u_1, ..., u_m)$ *Pareto dominates* a point $v = (v_1, ..., v_m)$ if, and only if, $u_k \leq v_k \; \forall k \in \{1, \ldots, m\} \wedge \exists k \in \{1, \ldots, m\} : u_k < v_k$. We denote this relation by $u \succ_P v$. We can now define an efficient solution: a feasible solution $x^* \in \mathcal{X}$ is called *efficient* if there does not exist any other feasible solution $x \in \mathcal{X}$ such that $y(x) \succ_P y(x^*)$ (its image in objective space is called a non-dominated point). For the sake of simplicity we will use the dominance relation w.r.t. solutions as well, i.e. $x \succ_P x^* \Leftrightarrow y(x) \succ_P y(x^*)$. The set that contains all efficient solutions is called the efficient set and is denoted by $\mathcal{X}_E$ (its image in objective space is called *Pareto front* and denoted by $\mathcal{Y}_N$).

In this paper, we focus our study on interactive approaches for solving many-objective optimization problems (at least 3 objectives). Interactive methods have been applied to solve multi-objective problems since 1971 [4]. The methods differ generally from the model used to integrate preferences of the DM. According to Miettinen *et al* [29], there are three types of specifying preference information in interactive methods, which are based on:

- Trade-off information: preferences are built on the quantities that a DM agrees to give up for a certain objective in order to improve another one to a certain quantity.
- Reference points: the DM needs to set a reference point (aspiration and reservation levels for all objective functions) and an achievement function is optimized.
- Classification of objective functions: the DM indicates which objectives should be improved, which one are acceptable as such and which ones are allowed to impair.

We focus our study on imprecise trade-offs, defined in the next section, that are a very convenient and fast way to integrate preferences of the DM.

3 Imprecise trade-offs

We assume that we have learned some preferences from the DM, that is there exists a set Θ of pairs of the form (u, v) meaning that the DM prefers the point u to the point v. We also assume the standard pointwise arithmetic operations, namely $u + v = (u_1 + v_1, \ldots, u_m + v_m)$ and $q \times u = (q \times u_1, \ldots, q \times u_m)$, where q is a real-value scalar. We are interested in building a new partial preference relation, called $\succeq_\Theta$, that extends Θ and the Pareto dominance $\succeq_P$. This new relation needs to satisfy the following two monotonicity properties, where $u, v, w \in \mathbb{R}^m$ are arbitrary vectors:

- **Independence:** if $u \succeq v$ then $u + w \succeq v + w$;
- **Scale-Invariance:** if $u \succeq v$ and $a \in \mathbb{R}, a \geq 0$ then $a \times u \succeq a \times v$.

This naturally leads to the following definitions [26]:

- We say that Θ is *consistent* if there exists some partial order $\succeq_\Theta$ that extends Θ, extends Pareto, and satisfies Scale-Invariance and Independence.
- For consistent Θ, we define the induced preference relation $\succeq_\Theta$ on $\mathbb{R}^p$ by $u \succeq_\Theta v \Leftrightarrow u \succeq v$ for all partial orders $\succeq$ such that $\succeq$ extends Pareto and Θ, and satisfies Independence and Scale-Invariance. This relation is a partial order as well.

Note that, as shown by Marinescu *et al* [27], the only class of utility functions that adheres the axioms of independence and scale-invariance are linear functions. Consequently, if a vector u is preferred to a vector v according to the preferences relation Θ, it also means that there exists some weight sets λ such that $\sum_i^m \lambda_i u_i \leq \sum_i^m \lambda_i v_i$ (with m objectives to minimize).

Example 1. *For example, let us consider a set $\mathcal{Y}$ composed of the following non-dominated points: $\mathcal{Y} = \{(5, 18), (7, 12), (12, 8), (17, 4)\}$. Let us consider that the DM prefers to reduce the first objective of one unit compared to reducing the second objective of one unit, that is $(-1, 0)$ is preferred to $(0, -1)$ and thus $\Theta = \{\big((-1, 0), (0, -1)\big)\}$. The independence property implies that $(0, 0) \succ_\Theta (1, -1)$ and also, for example, $(7, 12) \succ_\Theta (8, 11) \succ_\Theta (9, 10) \succ_\Theta \ldots \succ_\Theta (12, 7)$, and thus $(7, 12) \succ_\Theta (12, 8)$ since $(12, 7) \succ_P (12, 8)$. We have also $(7, 12) \succ_\Theta (17, 2)$ and thus $(7, 12) \succ_\Theta (17, 4)$ since $(17, 2) \succ_P (17, 4)$. Considering the new preference set Θ, the non-dominated set, called $\mathcal{Y}_\Theta$, is thus equal to $\{(5, 18), (7, 12)\}$.*

Given a set Θ, dominance check between two vectors can be done with linear programming, as done by Marinescu *et al* [26]. But in [27], Marinescu *et al* show that it is more efficient to use the algorithm of Tamura [32], based on the generators of the polyhedral cone defined by Pareto dominance and the preferences expressed in Θ. Using this algorithm, we obtain a matrix A of preferences that represent $\succeq_\Theta$ in the sense that $u \succeq_\Theta v$ only and only if $A(u - v) \succeq 0$ (this is if and only if $Au \succeq Av$).

Example 2. *For example, with the preferences of Example 1, by applying the Tamura algorithm*[2] *we obtain the following matrix:*

$$A = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$

Therefore we have $(7, 12) \succ_\theta (12, 8)$ since $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} -5 \\ 4 \end{pmatrix} = \begin{pmatrix} -5 \\ -1 \end{pmatrix}$ and $(-5, -1) \succ_P (0, 0)$. We have also $(7, 12) \succ_\Theta (17, 4)$ since $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} -10 \\ 8 \end{pmatrix} = \begin{pmatrix} -10 \\ -2 \end{pmatrix}$ and $(-10, -2) \succ_P (0, 0)$.

Also, as stated by Wiecek [33], each line of the matrix A corresponds to the weights of weighted sum (WS) functions to optimize. The number of lines of the matrix A (named q) represents the number of WS functions considered, and Pareto dominance among the values obtained for these different WS functions can be used to compare solutions with the dominance relation $\succ_\Theta$. Consequently, instead of computing the values obtained by the different WS functions each time that a dominance check following $\succ_\Theta$ is needed, we can simply

[2] We thank Radu Marinescu for sharing us their code of the Tamura algorithm.

store the evaluations of the projections of the solutions to the "weight space", that is the space produced by the different weight functions, and use Pareto dominance in this weight space. Note that the dimension of the weight space can be higher than the dimension of the original objective space. The problem of finding the efficient solutions according to the preference relation $\succeq_\Theta$ (called Θ-efficient solutions) becomes then equivalent to finding the Pareto efficient solutions of the WS-objective problems obtained from the matrix A (each line of A represents a WS objective).

Example 3. *For the preferences of Example 1, we have two WS objectives:* $fws_1(x) = y_1(x)$ *and* $fws_2(x) = y_1(x) + y_2(x)$*, and so the projections* $\mathcal{Y}$ *in the weight space* (fws_1, fws_2) *give* $\mathcal{Y}_{ws} = \{(5, 23), (7, 19), (12, 20), (17, 21)\}$*. If we keep only the Pareto non-dominated points from this set we obtain* $\mathcal{Y}_\Theta = \{(5, 18), (7, 12)\}$ *since we have* $(7, 19) \succ_P (12, 20)$ *and* $(7, 19) \succ_P (17, 21)$

The WS-objective functions, called $fw(x)_i$, can be written as follows:

$fws_i(x) = \sum_{j=1}^{m} a_{ij} y_j(x)\ \forall i = 1, \ldots, q$ where q is the number of WS-objective functions and a_{ij} are the elements of the matrix A.

4 Pareto local search with imprecise trade-offs

We present in this section how we have adapted a heuristic based on PLS to find a good approximation of the Θ-efficient solutions of multi-objective optimization problems. An approximation of a Θ-efficient set is a set of feasible solutions such that any pair of solutions of $\widehat{\mathcal{X}}_E^\Theta$ are mutually non-dominated, that is $\forall x \in \widehat{\mathcal{X}}_E^\Theta, \nexists x' \in \widehat{\mathcal{X}}_E^\Theta \mid x' \succeq_\Theta x$. This set will be also called a set of Θ-*potentially* optimal solutions.

4.1 Presentation

PLS [30] is an adaptation of simple local search based on improving moves to multi-objective optimization. PLS works directly with the current approximation of the efficient set. For each solution in the approximation its neighborhood is searched to find new potentially efficient solutions and to update the approximation.

There are slightly different versions of PLS according to the way the neighborhood is applied. There are versions where the neighborhood is only applied from non-dominated solutions [30] and there are versions where it is allowed to apply the neighborhood from dominated solutions [1, 17]. We will use here a version where the neighborhood can be applied from dominated solutions. The potentially efficient set obtained will be called $\widehat{\mathcal{X}}_E^\Theta$ and its image in objective space $\widehat{\mathcal{Y}}_N^\Theta$. The general algorithm of our version of PLS works as follows (see Algorithm 1). Two parameters are needed: an initial population $\mathcal{P}$ and a general preference relation $\succ_\Theta$ used to compare solutions (usually Pareto dominance). The population can be for example obtained by solving weighted sum problems.

The approximation $\widehat{\mathcal{X}}_E^\Theta$ is first updated with the population $\mathcal{P}$, that is only the potentially efficient solutions are kept in $\widehat{\mathcal{X}}_E^\Theta$. The procedure to update a non-dominated set $\widehat{\mathcal{X}}_E^\Theta$ is given in Algorithm 2 and simply consists in adding a new solution x to $\widehat{\mathcal{X}}_E^\Theta$ if it is non-dominated with respect to any solution in $\widehat{\mathcal{X}}_E^\Theta$ and to remove all solutions of $\widehat{\mathcal{X}}_E^\Theta$ that could be found dominated by x. Then, neighbors p' of the solutions p of $\mathcal{P}$ are generated. If a neighbor p' is not weakly dominated by the current solution p, we update a local list $\mathcal{L}_n$ of non-dominated neighbors with p'. Once all the neighbors p' of p have been generated, we update $\widehat{\mathcal{X}}_E^\Theta$ with all solutions of $\mathcal{L}_n$. All the new non-dominated solutions are then added to $\mathcal{P}$ for further exploration. We use a local list of non-dominated neighbors rather than updating $\widehat{\mathcal{X}}_E^\Theta$ each time that a new neighbor is produced (as commonly done in PLS, see [1, 23, 30]) in order to prefilter the candidate solutions and to keep only solutions of relatively good quality. In this way, the computational time needed to update $\widehat{\mathcal{X}}_E^\Theta$ can be reduced.

The neighbors list $\mathcal{L}_n$ is reinitialized after each neighborhood exploration. We remove then the solution p from the population and we start again the process until $\mathcal{P}$ is empty. Please note that in this version the population $\mathcal{P}$ may contain dominated solutions since we do not update $\mathcal{P}$ once a new solution is added. Indeed, it can be time-consuming to update $\mathcal{P}$ whose size can be quite large. Moreover keeping dominated solutions can improve the final results, as mentioned by Inja *et al* [20].

Algorithm 1 `PLS`

Parameter $\downarrow$: $\mathcal{P}$ (initial population of solutions), $\succ_\Theta$ (preference relation)
Parameter $\uparrow$: $\widehat{\mathcal{X}}_E^\Theta$ (approximation of the efficient set)

for all $p \in \mathcal{P}$ **do**
  `Update`($\widehat{\mathcal{X}}_E^\Theta$ $\updownarrow$,p $\downarrow$)
while $P \neq \emptyset$ **do**
  -- First solution of the population
  $p \leftarrow \mathcal{P}(0)$
  -- Generation of neighbors p'
  for all $p' \in \mathcal{N}(p)$ **do**
    if $(p \not\succeq_\Theta p')$ **then**
      `Update`($\mathcal{L}_N$ $\updownarrow$,p' $\downarrow$,$\succ_\Theta\downarrow$)
  for all $p' \in \mathcal{L}_N$ **do**
    if `Update`($\widehat{\mathcal{X}}_E^\Theta$ $\updownarrow$,p' $\downarrow$,$\succ_\Theta\downarrow$) **then**
      $\mathcal{P} \leftarrow \mathcal{P} \cup \{p'\}$
  $\mathcal{P} \leftarrow \mathcal{P} \setminus \{p\}$
  $\mathcal{L}_N \leftarrow \emptyset$

where $\mathcal{N}(p)$ denotes neighborhood of p, `Update()` updates a non-dominated set with a new solution and returns true if the new solution has been added to this set.

Algorithm 2 `Update`

IN $\downarrow$: x (solution), $\succ_\Theta$ (preference relation)
IN-OUT $\updownarrow$: $\widehat{\mathcal{X}}_E^\Theta$ (set of potentially efficient solutions)
OUT $\uparrow$: Boolean
if $(\nexists x' \in \widehat{\mathcal{X}}_E^\Theta \mid x' \succeq_\Theta x)$ **then**
  $\widehat{\mathcal{X}}_E^\Theta \leftarrow \widehat{\mathcal{X}}_E^\Theta \cup \{x\}$
  for all $(x' \in \widehat{\mathcal{X}}_E^\Theta \mid x \succ_\Theta x')$ **do**
    $\widehat{\mathcal{X}}_E^\Theta \leftarrow \widehat{\mathcal{X}}_E^\Theta \setminus \{x'\}$
  Return **True**
else
  Return **False**

4.2 Integration of trade-offs

We call I-PLS the integration of trade-offs into PLS. The pseudo-code of I-PLS is given in Algorithm 3. The main difference compared to the classic version of PLS is that the DM needs to set the maximal number of non-dominated solutions to generate ($maxS$) before running the algorithm. Then once more than $maxS$ solutions have been generated

by I-PLS, we ask the DM to compare two solutions in order to introduce some trade-offs between the objectives. Note that this operation is not without consequences for the performance of I-PLS. Indeed, the classic version of PLS uses the vast amount of information contained in the whole set of potentially Pareto optimal solutions to generate new solutions through the neighborhood function. By limiting the number of solutions in I-PLS, we will then also limit the quantity of information available and some adaptations will be necessary to attain high-quality results.

The detailed pseudo-code is given in Algorithm 3.

Algorithm 3 `I-PLS`

IN $\downarrow$: $maxS$ (maximum size)
OUT $\uparrow$: $\widehat{\mathcal{X}}_E^{\Theta}$ (set of potentially efficient solutions)

--| Initialization of a population P and $\widehat{\mathcal{X}}_E^{\Theta}$ with solutions of WS problems
`PopGeneration`($maxS \downarrow$,$\succ_\Theta \downarrow$,0 $\downarrow$, $\mathcal{P} \updownarrow$, $\widehat{\mathcal{X}}_E^{\Theta} \updownarrow$)
--| Initialization of an auxiliary population $\mathcal{P}_a$
$\mathcal{P}_a \leftarrow \emptyset$
while $\mathcal{P} \neq \emptyset$ **do**
  for all $p \in \mathcal{P}$ **do**
    --| Generation of neighbors p' of the solutions $p \in \mathcal{P}$
    for all $p' \in \mathcal{N}(p)$ **do**
      if $(p \not\succeq p')$ **then**
        if `Update`($\widehat{\mathcal{X}}_E^{\Theta} \updownarrow$,$p' \downarrow$,$\succ_\Theta \downarrow$) **then**
          $\mathcal{P}_a \leftarrow \mathcal{P}_a \cup \{p'\}$
  if ($|\widehat{\mathcal{X}}_E^{\Theta}| > maxS$) **then**
    while ($|\widehat{\mathcal{X}}_E^{\Theta}| > maxS$) **do**
      --| We ask the DM to add some preferences
      `UpdatePreferences`($\Theta \updownarrow, \widehat{\mathcal{X}}_E^{\Theta} \downarrow$)
      --| We update $\widehat{\mathcal{X}}_E^{\Theta}$ according to the additional preferences
      `UpdateXE`($\widehat{\mathcal{X}}_E^{\Theta} \updownarrow, \Theta \downarrow$)
    --| Generation of solutions of WS problems with the new preferences
    `PopGeneration`($maxS \downarrow$,$\succ_\Theta \downarrow$,0.05 $\downarrow$, $\mathcal{P}_a \updownarrow$, $\widehat{\mathcal{X}}_E^{\Theta} \updownarrow$)
  --| $\mathcal{P}$ is composed of the solutions of $\mathcal{P}_a$
  $\mathcal{P} \leftarrow \mathcal{P}_a$
  --| Reinitialization of $\mathcal{P}_a$
  $\mathcal{P}_a \leftarrow \emptyset$
--| Application of a Direct PLS
`D-PLS`($\Theta \updownarrow, \widehat{\mathcal{X}}_E^{\Theta} \updownarrow, maxS \downarrow$)

We first generate the initial population, with the `PopGeneration` algorithm, given in Algorithm 4. The initial population is composed of solutions optimizing WS problems. We first generate a random weight set and solve the WS problem obtained with a heuristic, to get a solution x. The solution x is then added to the initial population and the set of potentially efficient solution $\widehat{\mathcal{X}}_E^{\Theta}$ is updated with x. We stop the population generation when the maximal size has been reached or when a maximal number of iterations without improvement has been achieved. For the uniform random generation of the weight sets over the weight space defined by $\sum_i^m \lambda_i = 1, \lambda_i \geq 0$, we have followed the procedure developed by Rubinstein [31] that guarantees that the weight sets are uniformly generated. The Rubinstein procedure works as follows:

- Generation of $m-1$ independent random values $v_1, v_2, \ldots, v_{m-1} \in]0,1]$ from a uniform distribution
- Sorting the $m-1$ values: $1 \geq v_{(m-1)} \geq \ldots \geq v_{(2)} \geq v_{(1)} > 0$
- Generating the weight sets with the differences between the consecutive $v_{()}$ values:
  - $\lambda_m = 1 - v_{(m-1)}$
  - $\lambda_{m-1} = v_{(m-1)} - v_{(m-2)}$
  - $\ldots$
  - $\lambda_1 = v_{(1)} - 0$

Then we have that $\sum_{i=1}^m \lambda_i = 1$ and the weight sets are uniformly distributed (see [14] for a proof).

For generating random uniform weight sets when preferences have to be taken into account ($\Theta \neq \emptyset$), we work in the weight space defined by the q WS functions of the matrix A. We first generate q random weight w_i with the Rubinstein procedure. Then we use the transformation $\lambda_j = \sum_{i=1}^q w_i a_{ij}$ with $j = 1, \ldots, m$ (see end of Section 3) to obtain m random weight sets in the original space.

Algorithm 4 `PopGeneration`

IN $\downarrow$: $maxS$ (maximum size), $\succ_\Theta$ (preference relation), $perturb$ (amplitude of the perturbations)
IN-OUT $\updownarrow$: $\mathcal{P}$ (population), $\widehat{\mathcal{X}}_E^{\Theta}$ (set of potentially efficient solutions)
$cptWI \leftarrow 0$
while ($|\widehat{\mathcal{X}}_E^{\Theta}| < maxN$) and ($cptWI < 20$) **do**
  --| Generation of random weight sets
  $\lambda \leftarrow$ `WSGeneration`($\succ_\Theta$)
  --| Generation of a solution x with a heuristic method solving the WS problem
  $x \leftarrow$ `SolveWSHeuristic`($\lambda \downarrow$, perturb $\downarrow$)
  $\mathcal{P} \leftarrow \mathcal{P} \cup \{x\}$
  if `Update`($\widehat{\mathcal{X}}_E^{\Theta} \updownarrow$,$x \downarrow$,$\succ_\Theta \downarrow$) **then**
    $cptWI \leftarrow 0$
  else
    $cptWI \leftarrow cptWI + 1$

Note also that when solving the WS problems it is possible to introduce some data perturbations [11] (thanks to the $perturb$ parameter). Data perturbations are not used in the initial phase (the parameter $perturb$ is set to 0) of the algorithm but it will be useful in the second part of the algorithm, as it will be further detailed.

We then fully explore the neighborhood of the solutions of the population P, as done in the classic version of PLS. Each new potentially efficient solution is added to an auxiliary population, called $\mathcal{P}_a$, for further exploration. Then, at the end of the population exploration, we check if the number of potentially efficient solutions generated ($|\widehat{\mathcal{X}}_E^{\Theta}|$) is higher than $maxS$. If it is the case, we have to refine the Pareto dominance and to integrate some trade-offs, with the `UpdatePreferences` algorithm, described in Algorithm 5.

Algorithm 5 `UpdatePreferences`

IN $\downarrow$: $\widehat{\mathcal{X}}_E^{\Theta}$
IN-OUT $\updownarrow$: Θ
`FindClosestSolutions`($\widehat{\mathcal{X}}_E^{\Theta} \downarrow, y^1 \uparrow, y^2 \uparrow$)
$\Theta \leftarrow$ `Update`Θ($\Theta \downarrow, y^1 \uparrow, y^2 \uparrow$)

In this algorithm we ask the DM to compare two solutions of $\widehat{\mathcal{X}}_E^{\Theta}$ [5]. The two solutions to compare are carefully selected as follows. Among all the solutions of $\widehat{\mathcal{X}}_E^{\Theta}$, we select the two solutions that have the closest values over all the criteria except two. In this way,

the DM will have mainly to compare the solutions through only two criteria since for the other criteria the solutions will have closed values. More precisely, we need to find among all the points of $\widehat{\mathcal{Y}}_N^{\Theta}$ (image of $\widehat{\mathcal{X}}_E^{\Theta}$) the pair of points (y^1, y^2) that minimizes the following distance, called $dE2$:

$$dE2(y^1, y^2) = \min_{\{k,l\} \in \mathcal{M}, k \neq l} \max_{i \in \mathcal{M} \setminus \{k,l\}} |y_i^1 - y_i^2|$$

where $\mathcal{M}$ is the set of criteria.

That is we try to find the pair of solutions that minimizes the maximum distance between their values on m objectives, except on two objectives (in the formula above these two objectives are noted k and l).

Example 4. *If we consider three solutions* $y^1 = (6, 10, 7, 15), y^2 = (6, 20, 7, 8)$ *and* $y^3 = (11, 10, 11, 12)$*, we have* $dE2(y^1, y^2) = 0$ *(*$k = 2, l = 4$*),* $dE2(y^1, y^3) = 3$ *(*$k = 1, l = 3$*) and* $dE2(y^2, y^3) = 4$ $(k = 1, l = 2)$*. We will thus ask the DM to compare the solutions* y^1 *and* y^2*. As these solutions have the same values for the objectives 1 and 3, (s)he only needs to decide if (s)he prefers the values (10,15) or (20,8) for the objectives 2 and 4.*

Using mainly different values on two criteria is made on purpose to keep the request simple to answer for the decision maker. It allows the decision maker to mainly focus on two criteria rather than asking him to compare solutions with very different values on all criteria. Note that after asking the decision maker to compare two solutions according to two criteria, there is a high probability that the next question will about two other criteria.

The algorithm of this procedure, called `FindClosestSolutions`, is given in Algorithm 6.

Algorithm 6 `FindClosestSolutions`

IN $\downarrow$: $\widehat{\mathcal{X}}_E^{\Theta}$
OUT $\uparrow$: y^1, y^2
$minD \leftarrow Inf$
for all $(x^1, x^2) \in (\widehat{\mathcal{X}}_E^{\Theta})^2$ **do**
  if $(dE2(y(x^1), y(x^2)) < minD)$ **then**
    $minD \leftarrow dE2(y(x^1), y(x^2))$
    $y^1 \leftarrow y(x^1)$
    $y^2 \leftarrow y(x^2)$

Once (s)he has decided which solution (s)he prefers, some trade-offs are thus introduced and added to the preference relation Θ (see Algorithm 7).

Algorithm 7 `Update`Θ

IN-OUT $\updownarrow$: Θ
IN $\downarrow$: y^1, y^2
--| We ask to DM to compare y^1 and y^2
if $(y^1 \succ y^2)$ **then**
  $\Theta \leftarrow \Theta \cup \{y^1 \succ y^2\}$
else
  $\Theta \leftarrow \Theta \cup \{y^2 \succ y^1\}$

We then need to update $\widehat{\mathcal{X}}_E^{\Theta}$ since some solutions of $\widehat{\mathcal{X}}_E^{\Theta}$ are now dominated following the new preference relation. This all process can be done several times until $|\widehat{\mathcal{X}}_E^{\Theta}| \leq maxS$. Then we generate some more solutions coming from WS problems. Indeed, as more preferences have been added, we can focus the search into a particular region of the objective space. We apply thus again the `PopGeneration` algorithm, at the exception that some data perturbations will be introduced to diversify the search and to maximize the probability to generate new potentially Θ-efficient solutions [11]. The data perturbation method simply works by giving a small perturbation of maximal 5% to the different data of the problem. For example, for the TSP, we will modify the m costs of the edges by increasing or decreasing them by maximum 5% of their initial values (different values have been tried and the experiments showed that 5% allows to obtain the best results).

Then we transfer the solutions of the auxiliary population $\mathcal{P}_a$ to $\mathcal{P}$ and we start again the exploration of the neighborhood from the solutions of $\mathcal{P}$. Please note that we do not update $\mathcal{P}_a$ even if new preferences have been added, in order to keep a diversified population, not only composed of Θ-efficient solutions.

Once no more improvements can be achieved by exploring the neighborhood of $\mathcal{P}$ ($\mathcal{P} = \emptyset$) and not more than $maxS$ solutions have been generated, we apply a *Direct* Pareto local search. The method is very similar to I-PLS, except that:

- Perturbations are introduced in the initial population generation (max 5%)
- In order to diversify the search, dominated neighbors can be added into $\mathcal{P}_a$: we will accept dominated neighbors with a probability decreasing with the number of times that the population has been explored. We start with a value of $\frac{1}{2}$ and then after each full population exploration, the probability is decreasing to $\frac{1}{3}$, $\frac{1}{4}$, etc.

5 Results

We have applied the new method I-PLS to the multi-objective traveling salesman problem (MOTSP). As to our knowledge no results are known for the MOTSP with up to 6 objectives, we will compare I-PLS with an *a posteriori* method: we have implemented an efficient version of PLS based on [11] where Pareto dominance is used. The potentially Pareto efficient set obtained at the end is then filtered according to the preference relation Θ. In both methods, WS problems are solved with one of the best heuristics for the single-objective TSP: the version of Applegate [2] of the Lin-Kernighan heuristic [22]. For the neighborhood simple 2-opt moves (two edges are removed, and two new edges are added) are considered. We have also used the new data structure ND-Tree [18] to efficiently manage and update the set of potentially efficient solutions. Note that ND-Tree can be easily adapated to I-PLS since by working in the weight space defined by the matrix A obtained from the set of preferences Θ, Pareto dominance can be employed and no changes in the ND-Tree structure and algorithm are needed. Finally, when a DM is asked to compare two solutions, the solution that s(he) prefers is randomly selected.

All the results have been obtained on a Intel Core i5-450M CPU, at 2.4 GHz[3]. We have used MOTSP instances with 3 to 6 objectives, and respectively 100, 30, 20 and 15 cities. We first used instances whose the number of cities is rather low with the aim to keep a reasonable number of potentially efficient solutions when PLS is used with the Pareto dominance relation. The m costs associated to each edge are randomly generated with a uniform distribution. The instances are symmetric, that is the cost for moving to a city i to a city j is equal to the cost for moving from the city j to the city i.

Please note that our aim is not to produce new state-of-the-art results for the MOTSP but to show that I-PLS can reach same quality

[3] The instances, code and results are available at `https://sites.google.com/site/interactivepls/`.

or better quality results than a Pareto approach in less computational time, for many-objective instances when trade-offs are considered.

The results obtained by PLS are given in Table 1. We see that the number of potentially efficient solutions generated is very high (e.g. more than 600 000 solutions for the 5 objectives instance) even if the number of cities is quite small.

Instance	$\lvert\widehat{\mathcal{X}}_E\rvert$	CPU(s)
3-100	153 114	185.79
4-30	265 794	254.70
5-20	690 607	2103.03
6-15	338 396	1219.76

Table 1. Results obtained by PLS for the small size instances.

We show now the results obtained by I-PLS by limiting the number of solutions generated to 1, 10 and 100 and compare to the results obtained by PLS. The results of PLS are obtained by filtering the potentially efficient solutions with the preference relation obtained at the end of I-PLS. We ignore the computational time of this operation, which is relatively small compared to the time needed to generate the potentially efficient solutions.

To compare PLS and I-PLS, we use three indicators:

- The average distance D_1 and maximum distance D_2 (to be minimised) [12] between the points of a reference set and the points of $\widehat{\mathcal{Y}}_N^{\Theta}$, by using the Euclidean distance:

$$D_1(\widehat{\mathcal{Y}}_N^{\Theta}, \mathcal{RS}) = \frac{\sum_{r \in \mathcal{RS}} \min_{y \in \widehat{\mathcal{Y}}_N^{\Theta}} d(r, y)}{|\mathcal{RS}|}$$

$$D_2(\widehat{\mathcal{Y}}_N^{\Theta}, \mathcal{RS}) = \max_{r \in \mathcal{RS}} \min_{y \in \widehat{\mathcal{Y}}_N^{\Theta}} d(r, y)$$

  where $\mathcal{RS}$ is a reference set, and $d(.,.)$ denotes Euclidean distance.
- The proportion PR of reference solutions found.

$$PR(\widehat{\mathcal{Y}}_N^{\Theta}, \mathcal{RS}) = \frac{|\mathcal{RS} \cap \widehat{\mathcal{Y}}_N^{\Theta}|}{|\mathcal{RS}|}$$

The reference sets are obtained by merging the sets obtained by PLS, I-PLS and the results obtained by a Direct PLS that uses the final preference relation Θ and a high quality population, composed of solutions generated by the exact TSP solver Concorde [3] and the Lin-Kernighan heuristic [16].

The results of the comparison between PLS and I-PLS are given in Table 2. We give the values of the maximal size $maxS$ parameter, the number |tofs| of trade-offs considered, the number $|\widehat{\mathcal{Y}}_N^{\Theta}|$ of solutions generated (always inferior to $maxS$), the distance D_1, the distance D_2, the proportion PR of reference solutions found and the CPU time in seconds.

We note that:

- The number of tofs is between 4 and 14. Its average for $maxS = 1$ is equal to 9.5, for $maxS = 10$ equal to 7.5, for $maxS = 100$ equal to 8.25. Note that for some instances, the number of tofs is increasing even if $maxS$ is also increasing; indeed, to each tof corresponds a comparison between two solutions and there are some comparisons that are more informative than others.

Instance	$maxS$	|tofs|	Method	$\lvert\widehat{\mathcal{Y}}_N^{\Theta}\rvert$	D_1	D_2	PR	CPU(s)
3-100	1	14	PLS	1	19.04	19.04	0.00	185.79
			I-PLS	1	**0.00**	**0.00**	**1.00**	**11.06**
4-30	1	9	PLS	1	**0.00**	**0.00**	**1.00**	254.70
			I-PLS	1	**0.00**	**0.00**	**1.00**	**0.39**
5-20	1	6	PLS	1	**0.00**	**0.00**	**1.00**	2101.03
			I-PLS	1	**0.00**	**0.00**	**1.00**	**0.27**
6-15	1	9	PLS	1	**0.00**	**0.00**	**1.00**	1219.76
			I-PLS	1	**0.00**	**0.00**	**1.00**	**2.02**
3-100	10	12	PLS	5	0.76	4.57	0.83	185.79
			I-PLS	6	**0.00**	**0.00**	**1.00**	**34.48**
4-30	10	7	PLS	10	**0.00**	**0.00**	**1.00**	254.70
			I-PLS	7	6.80	37.59	0.70	**0.68**
5-20	10	4	PLS	1	**0.00**	**0.00**	**1.00**	2101.03
			I-PLS	1	**0.00**	**0.00**	**1.00**	**0.22**
6-15	10	7	PLS	9	**0.00**	**0.00**	**1.00**	1219.76
			I-PLS	7	6.16	29.75	0.78	**7.52**
3-100	100	8	PLS	41	1.64	**5.35**	0.32	185.79
			I-PLS	60	**1.00**	11.78	**0.62**	**31.68**
4-30	100	11	PLS	88	1.16	17.48	0.88	254.70
			I-PLS	89	**0.88**	**14.83**	**0.88**	**45.91**
5-20	100	5	PLS	58	**0.00**	**0.00**	**1.00**	2101.03
			I-PLS	58	**0.00**	**0.00**	**1.00**	**46.79**
6-15	100	9	PLS	52	**0.00**	**0.00**	**1.00**	1219.76
			I-PLS	52	**0.00**	**0.00**	**1.00**	**57.54**

Table 2. Comparison between PLS and I-PLS for small size instances (the best values obtained are in bold).

- The CPU of I-PLS is always inferior to PLS: the speed-up is between 5 (3-100 instance with $maxS = 10$) and 9550 (5-20 instance with $maxS = 10$).
- The values of the indicators of I-PLS are always better or equal to those of PLS, except for the 4-30 instance with $maxS = 10$, the 6-15 instance with $maxS = 10$ and the 3-100 instance with $maxS = 100$ where PLS gets better results for the D_2 indicator. For these instances, we see that the running time of I-PLS is very small compared to PLS and I-PLS has probably stopped too early. Indeed, I-PLS should be able to behave like a multiobjective algorithm at the beginning, and more as a single-objective at the end. Probably that the intensification phase was not enough for these instances to get results of equal quality compared to PLS.

We have also experimented I-PLS on bigger instances: 200 cities with m going from 3 to 6. The maximal size of the set has been fixed to 10. We still compare I-PLS to PLS but to avoid prohibitive computational time we have interrupted PLS once that 500 000 potentially efficient solutions have been generated. The results are given in Table 3.

Instance	$maxS$	|tofs|	Method	$\lvert\widehat{\mathcal{Y}}_N^{\Theta}\rvert$	D_1	D_2	PR	CPU(s)
3-200	10	8	PLS	1	8.01	8.01	0.00	5245.67
			I-PLS	1	**0.00**	**0.00**	**1.00**	**531.36**
4-200	10	9	PLS	1	278.45	287.35	0.00	2161.02
			I-PLS	8	**4.54**	**26.56**	**0.70**	**92.98**
5-200	10	17	PLS	1	110.70	114.36	0.00	309.77
			I-PLS	3	**9.52**	**11.51**	**0.00**	**241.18**
6-200	10	14	PLS	1	72.64	81.22	0.00	1998.37
			I-PLS	8	**5.71**	**17.81**	**0.38**	**948.85**

Table 3. 200 cities instances, $maxS = 10$.

We remark that:

- PLS, after filtering, only generates one solution: we believe that is mainly because PLS generates many solutions in different regions of the objective space without focusing the search in particular regions as I-PLS.

- The values of the indicators D_1, D_2, PR of I-PLS are considerably better than the values obtained by PLS.
- I-PLS is still faster than PLS, with speed-ups between 1.3 and 23.

Compared to the smaller size instances, the number of questions asked is on average higher (12 against 7.5) but remains quite small if we consider that the number of Pareto efficient solutions is at least higher than 500 000, while I-PLS generates at the end less than 10 solutions. Also, the questions asked to the DM focus only on two objectives (see Section 4.2) while more informative questions (but also more complex) could be asked.

6 Conclusion

We have presented in this paper the integration of imprecise trade-offs into multi-objective optimization to obtain an interactive version of Pareto local search. We have applied the method for the first time to multi-objective instances of the TSP with up to 6 objectives and the results showed that the new method can reach high quality results in very short time compared to a Pareto approach. Moreover, the number of questions asked to the DM to elicitate his/her preferences is quite small (between 4 and 17 questions in our experiments). This work opens many new perspectives: our work mainly focus on Pareto local search but it would be interesting to see how other heuristics can deal with imprecise trade-offs. Also, adaptation of exact methods to find in an interactive way the preferred solutions of the DM will be worth to be studied. Finally, there are different possibilities to choose the solutions that are compared by the DM: it would be relevant to see in practical way if our approach is conceivable and to study other approaches, for example approaches that try to minimize the number of questions asked to the DM [9] or to take noisy comparisons into consideration [8].

REFERENCES

[1] E. Angel, E. Bampis, and L. Gourvès, 'A dynasearch neighborhood for the bicriteria traveling salesman problem', in *Metaheuristics for Multiobjective Optimisation*, eds., X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, 153–176, Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535, Berlin, (2004).

[2] D. Applegate, 'Chained Lin-Kernighan for large traveling salesman problems', *INFORMS Journal on Computing*, **15**, 82–92, (2003).

[3] D. Applegate, R. Bixby, V. Chvátal, and W. Cook, *TSP Cuts Which Do Not Conform to the Template Paradigm*, 261–303, Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[4] R. Benayoun, J. de Montgolfier, J. Tergny, and O. Laritchev, 'Linear programming with multiple objective functions: Step method (stem)', *Mathematical Programming*, **1**(1), 366–375, (1971).

[5] J. Branke, S. Corrente, S. Greco, and W. Gutjahr, 'Efficient pairwise preference elicitation allowing for indifference', *Computers & Operations Research*, **88**, 175 – 186, (2017).

[6] J. Branke and K. Deb, *Integrating User Preferences into Evolutionary Multi-Objective Optimization*, 461–477, Knowledge Incorporation in Evolutionary Computation, Springer Berlin Heidelberg, 2005.

[7] K. Bringmann, T. Friedrich, F. Neumann, and M. Wagner, 'Approximation-guided evolutionary multi-objective optimization', in *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, ed., Toby Walsh, pp. 1198–1203. IJCAI/AAAI, (2011).

[8] U. Chajewska, D. Koller, and R. Parr, 'Making rational decisions using adaptive utility elicitation', in *In Proceedings of the Seventeenth National Conference on Artificial Intelligence*, pp. 363–369, (2000).

[9] K. Ciomek, M. Kadziński, and T. Tervonen, 'Heuristics for prioritizing pair-wise elicitation questions with additive multi-attribute value models', *Omega*, **71**, 27 – 45, (2017).

[10] V.N. Coelho, M.J.F. Souza, I.M. Coelho, F.G. Guimarães, T. Lust, and R.C. Cruz, 'Multi-objective approaches for the open-pit mining operational planning problem', *Electronic Notes in Discrete Mathematics*, **39**, 233–240, (2012).

[11] M. Cornu, T. Cazenave, and D. Vanderpooten, 'Perturbed decomposition algorithm applied to the multi-objective traveling salesman problem', *Computers & Operations Research*, **79**, 314 – 330, (2017).

[12] P. Czyzak and A. Jaszkiewicz, 'Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization', *Journal of Multi-Criteria Decision Analysis*, **7**, 34–47, (1998).

[13] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, 'A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II', in *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pp. 849–858, Paris, France, (2000). Springer. Lecture Notes in Computer Science No. 1917.

[14] L. Devroye, 'Sample-based non-uniform random variate generation', in *Proceedings of the 18th Conference on Winter Simulation*, WSC '86, pp. 260–265, New York, NY, USA, (1986). ACM.

[15] J.-P. Dubus, C. Gonzales, and P. Perny, 'Multiobjective optimization using GAI models', in *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, ed., C. Boutilier, pp. 1902–1907, (2009).

[16] K. Helsgaun, 'An effective implementation of the Lin-Kernighan traveling salesman heuristic', *European Journal of Operational Research*, **126**, 106–130, (2000).

[17] M. Inja, C. Kooijman, M. De Waard, D. Roijers, and S. Whiteson, 'Queued Pareto local search for multi-objective optimization', in *PPSN 2014: Proceedings of the Thirteenth International Conference on Parallel Problem Solving from Nature*, pp. 589–599, (September 2014).

[18] A. Jaszkiewicz and T. Lust, 'ND-Tree-based update: a fast algorithm for the dynamic non-dominance problem', *IEEE Transactions on Evolutionary Computation*, 15, (2018).

[19] E. Karasakal and M. Köksalan, 'Generating a representative subset of the nondominated frontier in multiple criteria decision making', *Operations Research*, **57**(1), 187–199, (2009).

[20] C. Kooijman, M. De Waard, M. Inja, D. Roijers, and S. Whiteson, 'Pareto local policy search for MOMDP planning', in *ESANN 2015: Proceedings of the 23rd European Symposium on Artificial Neural Networks, Special Session on Emerging Techniques and Applications in Multi-Objective Reinforcement Learning*, pp. 53–58, (April 2015).

[21] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, 'Combining convergence and diversity in evolutionary multiobjective optimization', *Evol. Comput.*, **10**(3), 263–282, (September 2002).

[22] S. Lin and B.W. Kernighan, 'An effective heuristic algorithm for the traveling-salesman problem', *Operations Research*, **21**, 498–516, (1973).

[23] T. Lust and J. Teghem, 'Two-phase Pareto local search for the biobjective traveling salesman problem', *Journal of Heuristics*, **16**(3), 475–510, (2010).

[24] T. Lust and J. Teghem, 'The multiobjective multidimensional knapsack problem: a survey and a new approach', *International Transactions in Operational Research*, **19**(4), 495–520, (2012).

[25] T. Lust and D. Tuyttens, 'Variable and large neighborhood search to solve the multiobjective set covering problem', *J. Heuristics*, **20**(2), 165–188, (2014).

[26] R. Marinescu, A. Razak, and N. Wilson, 'Multi-objective influence diagrams', in *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, August 14-18, 2012*, eds., N. De Freitas and K.P. Murphy, pp. 574–583. AUAI Press, (2012).

[27] R. Marinescu, A. Razak, and N. Wilson, *Principles and Practice of Constraint Programming: 19th International Conference, CP 2013, Uppsala, Sweden, September 16-20, 2013. Proceedings*, chapter Multi-Objective Constraint Optimization with Tradeoffs, 497–512, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[28] K. Miettinen, *Interactive Nonlinear Multiobjective Procedures*, 227–276, Springer US, Boston, MA, 2002.

[29] K. Miettinen, F. Ruiz, and A.P. Wierzbicki, *Introduction to Multiobjective Optimization: Interactive Approaches*, 27–57, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[30] L. Paquete, M. Chiarandini, and T. Stützle, 'Pareto local optimum sets in the biobjective traveling salesman problem: an experimental study', in *Metaheuristics for Multiobjective Optimisation*, eds., X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, pp. 177–199, Berlin, (2004). Springer. Lecture Notes in Economics and Mathematical Systems Vol.

535.
[31] R.Y. Rubinstein, 'Generating random vectors uniformly distributed inside and on the surface of different regions', *European Journal of Operational Research*, **10**(2), 205 – 209, (1982).
[32] K. Tamura, 'A method for constructing the polar cone of a polyhedral cone, with applications to linear multicriteria decision problems', *Journal of Optimization Theory and Applications*, **19**(4), 547–564, (1976).
[33] M. M. Wiecek, 'Advances in cone-based preference modeling for decision making with multiple criteria', *Decision Making in Manufacturing and Services*, **Vol. 1, no. 1-2**, 153–173, (2007).
[34] N. Wilson, A. Razak, and R. Marinescu, 'Computing possibly optimal solutions for multi-objective constraint optimisation with tradeoffs', in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, eds., Q. Yang and M. Wooldridge, pp. 815–822. AAAI Press, (2015).