A genetic-based framework for solving (multi-criteria) weighted matching problems

Andrés L. Medaglia a, Shu-Cherng Fang b,*

a Departamento de Ingeniería Industrial, Facultad de Ingeniería, Universidad de los Andes, AA 4976, Bogotá DC, Colombia
b Department of Industrial Engineering and Graduate Program in Operations Research, North Carolina State University, Raleigh, NC 27695-7906, USA

Received 25 February 2002; accepted 9 May 2002

Abstract

The purpose of this paper is to present a flexible genetic-based framework for solving the multi-criteria weighted matching problem (mc-WMP). In the first part of this paper, we design a genetic-based framework for solving the ordinary weighted matching problem. We present an extensive analysis of the quality of the results and introduce a methodology for tuning its parameters. In the second part, we develop a modified genetic-based algorithm for solving the mc-WMP. The algorithm generates a significant and representative portion of the Pareto optimal set. To assess the performance of the algorithm, we conduct computational experiments with two and three criteria. The potential of the proposed algorithm is demonstrated by comparing to a multi-objective simulated annealing algorithm.

Keywords: Genetic algorithms; Multi-objective evolutionary algorithms; Multi-criteria optimization; Weighted matching

1. Introduction

Let \( G = (V, E) \) be an undirected graph with \( V \) and \( E \) being its node set and edge set, respectively. The graph \( G \) is complete if every pair of nodes is linked by an edge. A subset \( M \subseteq E \) is called a matching if no two edges in \( M \) are incident to the same node. A node in \( V \) is said to be matched or covered, if there is an edge in \( M \) incident to it. An exposed node is one that is not covered. A matching that leaves no exposed nodes is called a perfect matching.

Given a graph \( G = (V, E) \) and weights \( w_{ij} \) associated with each edge \( (i, j) \in E \), the maximum (minimum) weighted matching problem (WMP) finds a matching in \( G \) with the largest (smallest) possible sum of weights. Applications of the WMP include routing of mechanical plotters [41], scheduling of crews and vehicles [5], scheduling of training sessions [6], and vehicle routing [14,38].

The work of Edmonds [15] on matching is considered one of the most important contributions in
the area of combinatorial optimization. He was able to prove that the polyhedron associated with the weighted matching problem has 0–1 vertices and proposed a very efficient polynomial time algorithm, namely the Blossom Algorithm. The complexity of the algorithm is of $O(|V|^2|E|)$, where $|V|$ is the number of nodes and $|E|$ is the number of edges in the graph. Several researchers have been improving this bound ever since [22,24]. Gabow’s bound [21] of $O(|V|(|E| + |V| \log |V|))$ is the best bound known to date based exclusively on the number of nodes and edges. Better bounds can be obtained if the edge weights are of integer value [23].

Many versions of the Blossom Algorithm have been implemented [7,10,12,13,36,38]. Applegate and Cook [1] presented an implementation for solving large scale WMPs with up to 101,230 nodes. More recently, Cook and Rohe [9] reported the successful solution of instances with up to 5,000,000 nodes.

Modifications made to the original formulation of this polynomially solvable problem can lead to the realm of NP-hardness, for example, the maximum three-dimensional matching [33], maximum subgraph matching [25], minimum maximal matching [25], and the multi-criteria weighted matching [25], and the multi-criteria weighted matching [33]. Actually, Ulungu et al. [46] consider that by the standard of the classical NP-hard complexity of most combinatorial problems, multi-criteria combinatorial optimization problems are particularly ‘intractable’ by using an ‘exact’ method. The purpose of this paper is to present a genetic-based framework which is flexible for solving the multi-criteria weighted matching problem (mc-WMP).

The paper is organized as follows. Section 2 presents a basic framework for solving the ordinary WMP. Extensive computational results are discussed, and a methodology for tuning parameters is included. In Section 3, the framework is extended for solving the mc-WMP. This section also presents computational experiments with two and three criteria and a comparison to a multi-objective simulated annealing algorithm (MOSA). Concluding remarks and future research are given in Section 4. Two appendices contain the detailed results presented in Section 2.

2. Genetic-based approach

2.1. Weighted matching problem

In this paper, we consider an undirected graph $G = (V,E)$ with weights $w_{ij} \geq 0$ for each $(i,j) \in E$. For simplicity and without loss of generality, we assume that the graph is complete with an even number of nodes.\(^1\)

Let $x_{ij} = 1$, if node $i$ is matched to node $j$ in matching $M$; $x_{ij} = 0$, otherwise. The matching can be written as $M = \{ (i,j)|x_{ij} = 1, (i,j) \in E \}$.

The maximum (minimum) weighted perfect matching problem is to find a perfect matching $M$ of $G$ such that $\sum_{(i,j) \in M} w_{ij}$ is maximized (minimized). As in the literature, we call this problem the WMP. For simplicity, we focus on the maximization case, but it could also be applied to the minimization case with some minor modifications.

2.2. Framework

Evolutionary algorithms are stochastic search techniques based on the principles of natural selection. There have been several schools of thought that have contributed and enriched the field, but share the same underlying principles (i.e., evolutionary strategies [40,42], evolutionary programming [17], and genetic algorithms [30]).

The basic idea of a genetic-based algorithm is to simulate an evolutionary process and obtain better individuals (i.e., solutions) as the algorithm progresses. In every generation (i.e., iteration) the solutions are combined and altered through the so-called crossover and mutation operators. Through the evaluation and selection of the candidate solutions (i.e., individuals) the fitter ones have better chance of survival. In this search process, good solutions can be obtained and local optima can be avoided. Genetic-based algorithms have been applied to a variety of problems, especially those for which conventional methods fail to give an

\(^1\) In the case that the graph is not complete, we can let the weights of those edges that are missing in $G$ be equal to zero. If there is an odd number of vertices, we can always add a new node with edges of weight zero incident upon it.
Genetic-based algorithm:

$\begin{align*}
& t \leftarrow 1 \\
& \text{Initialize } P(t) \\
& \text{Evaluate } P(t) \\
& \text{While } t \leq N \text{ do} \\
& \quad \text{Recombine and vary } P(t) \text{ and generate } C(t) \\
& \quad \text{Evaluate } C(t) \\
& \quad \text{Select } P(t+1) \text{ from } P(t) \text{ and } C(t) \\
& \quad t \leftarrow t + 1
\end{align*}$

where $t$ is the generation (iteration) index, $N$ is the maximum number of generations, $P(t)$ is the population of solutions in the generation $t$, and $C(t)$ is the children population in generation $t$.

### 2.3. Encoding

Choosing an appropriate representation of the candidate solutions is a key feature in any genetic-based algorithm. Let us consider the matrix representation of a matching $M$ depicted in Fig. 1. If the edge $(i, j)$ is in the matching $M$, a 1 is placed in the cell located at the intersection of row $i$ and column $j$. For feasibility, given that node $i$ is already covered, no more assignments should be made on the row or column associated with this node. Note that for an undirected graph, only the upper triangular portion of the matrix is used. Fig. 1 is a matrix representation of the graph shown in Fig. 2.

Even though the matrix is a suitable and intuitive way of representing a matching, it is by no means an efficient way. We exploit the benefits of the matrix representation without storing the full matrix, but a list of the selected edges. In this way, instead of storing $(|V|(|V| - 1))/2$ elements, only $(1/2)|V|$ elements are contained in the list.

### 2.4. Generating initial population

The procedure presented in this section generates feasible matchings for the initial population. The basic idea of this procedure is to construct feasible matchings by picking valid edges progressively. In general, edges with better weights are most likely to be picked.

Let $R$ and $C$ be the sets of available “rows” (nodes) and “columns” (nodes) to pick from to form a new matching, respectively. Let $M$ be the matching generated by the procedure.

**Step 0:** Initializing.

Set $R \leftarrow \{1, 2, \ldots, |V| - 1\}$, $C \leftarrow \{2, 3, \ldots, |V|\}$, $M \leftarrow \emptyset$, $\overline{R} \leftarrow R$, and $\overline{C} \leftarrow C$.

**Step 1:** Terminating.

If $R = \emptyset$ and $C = \emptyset$, then STOP.

**Step 2:** Picking random edges.

For each $i \in R$ and $j \in C$, let $\delta_{ij} = 1$, if $i < j$;
\[ \delta_{ij} = 0, \text{ otherwise.} \]

For each \( i \in R \), let the probability of selecting node \( i \) be 
\[ p_i = \frac{\delta_{ij}}{\sum_{j \in C} \delta_{ij}} \]
if \( i \in R \), \( p_i = 0 \), otherwise. Also define the cumulative probability 
\[ g_i = \sum_{r \in R, r < i} f_r. \]
Set \( g_0 = 0 \).

Generate a random variate \( u \) from a uniform distribution over the range \((0, 1]\). Pick the \( i \)'th row (node) such that \( g_{r-1} < u \leq g_r \).

Let \( C' = \{ j | j > i; j \in C \} \). With the convention of \( 0/0 = 0 \), for each \( j \in C \), the probability of choosing node \( j \) given that node \( i \) has been chosen, is given by 
\[ p_{rj} = \frac{w_{rj}}{\sum_{c \in C'} w_{c,i}} \]
if \( j \in C' \); \( p_{rj} = 0 \), otherwise. Also define 
\[ q_j = \frac{w_{i,j}}{\sum_{c \in C} w_{c,i}} \]
Set \( q_1 = 0 \).

Generate a random variate \( u \) from a uniform distribution over the range \((0, 1]\). Pick the \( j \)'th column (node) such that 
\[ q_{j-1} < u \leq q_j. \]
If \( \sum_{j \in C} w_{rj} = 0 \), then pick the \( j \)'th node randomly from \( C' \).

**Step 3:** Updating.
\[ R \leftarrow R \setminus \{i', j\}, \quad C \leftarrow C \setminus \{i', j\}, \quad \text{and} \quad M \leftarrow M \cup \{(i', j)\}. \]
Go to Step 1.

### 2.5. Generating new population

The purpose of this procedure is to complete the construction of a matching that has been partially created by the mutation and/or crossover operators.

**Step 0:** Initializing.

Let \( R, C, \) and \( M \) be the respective sets for the available rows, available columns and the matching, after undergoing the mutation or crossover operator.

**Step 1:** Terminating.

If \( R = \emptyset \) and \( C = \emptyset \), then STOP.

**Step 2:** Picking a random edge \((i', j)\).

Pick \( i' \in R \) and \( j \in C \) such that \( i' < j \).

**Step 3:** Updating.
\[ R \leftarrow R \setminus \{i', j\}, \quad C \leftarrow C \setminus \{i', j\}, \quad \text{and} \quad M \leftarrow M \cup \{(i', j)\}. \]
Go to Step 1.

### 2.6. Evaluation and selection

The principle behind a genetic-based algorithm is the natural selection. An appropriate selection pressure guarantees the balance between exploration of the search space and the exploitation of the good candidate solutions.

The fitness of each individual is a value associated with the encoded solution. In our case, given a matching \( M \) (individual), its fitness is obtained by adding the weights of the edges in \( M \). In other words, each individual \( M \) of the population is evaluated by assigning its fitness to be \( \sum_{(i,j) \in M} w_{ij} \).

In the proposed genetic-based algorithm, we have chosen to perform the selection on an enlarged sampling space. This method known as \((\mu + \lambda)\) selection was first introduced by Bäck and Hoffmeister [3]. In this type of selection, \( \mu \) parents \((P(i)) \) and \( \lambda \) children \((C(i)) \) compete for survival with the \( \mu \) best individuals being selected from the pool of size \( \mu + \lambda \).

#### 2.7. Genetic operators

##### 2.7.1. Crossover

The main feature of the crossover operation is to preserve the good genes of the two parents. The better edges of each parent have a larger chance of becoming part of a newly created child. Nevertheless, the probabilistic nature of this design leaves open the possibility that not-so-good edges might still be included.

As pointed out by Gen and Cheng [26], locality is a desired feature when dealing with a combinatorial problem. Locality has to be understood in the sense that small changes in the representation make small changes in the solution. With the procedure that we proposed below, the crossover operation produces a child that have a good chance of resembling its two parents.

Every individual in the population \( P(t) \), has a probability \( p_c \) (probability of crossover) of being chosen to form the crossover pool of parents. The procedure described in this section is applied to every pair of parents randomly selected from this pool. Furthermore, every individual in the crossover pool is allowed to be parent of at most one child.

Let \( R \) and \( C \) be the available rows and columns to pick from the existing matching to form a new one, respectively. For a pair of matchings randomly selected from the crossover pool, let \( M_1 \) be a
copy of parent 1 and $M_2$ be a copy of parent 2. The crossover operator will produce a child matching $M_3$.

**Step 0:** Initializing.

- $R \leftarrow \{1, 2, \ldots, |V| - 1\}$, $C \leftarrow \{2, 3, \ldots, |V|\}$,
- $M_3 \leftarrow \emptyset$, and $s \leftarrow 1$.

**Step 1:** Terminating.

If $M_s = \emptyset$, then go to Step 4.

**Step 2:** Picking an edge from a parent.

Let $e_m$ be the $m$th edge in $M_s$ with weight $w_{e_m}$. With the convention of $0/0 = 0$, let the probability of choosing edge $e_m$ be $p_{e_m} = w_{e_m} / \sum_{e \in M_s} w_e$. Define the cumulative probability $f_{e_m} = \sum_{e=1}^m p_e$, for $k = 1, \ldots, |M_s|$. Set $f_{e_m} = 0$.

Generate a random variate $u$ from a uniform distribution over the range $(0, 1]$. Pick the $m$th edge ($e_{m'} = (i', j')$) from $M_s$ such that $f_{e_{m'-1}} < u \leq f_{e_{m'}}$. If $\sum_{e \in M_s} w_e = 0$, then pick $e_{m'}$ randomly from $M_s$.

**Step 3:** Updating.

If $s = 1$ then $s \leftarrow 2$, else $s \leftarrow 1$. Set $R \leftarrow R \setminus \{(i', j')\}$ and $C \leftarrow C \setminus \{(i', j')\}$. Also set $M_e \leftarrow M_s \setminus \{(i, j)\}$ for $i \in \{i', j'\}$ and $j \in \{i', j'\}$ (for $v = 1, 2$) and $M_3 \leftarrow M_3 \cup \{(i', j')\}$. Go to Step 1.

**Step 4:** Generating the rest of the matching.

Go to the reconstruction procedure for generating new population (Section 2.5).

### 2.7.2. Mutation

The proposed mutation operator is basically a two-phase procedure. In the first phase, some edges are randomly eliminated from the individual, while the second phase is a reconstruction stage, where the lost feasibility is recovered.

Let $p_m$ be the probability of eliminating an edge, $R$ and $C$ be the sets of available rows (nodes) and columns (nodes) to pick from an existing matching $M$. Also set flag = 1 if at least one of the edges from $M$ has been eliminated; flag = 0, otherwise.

**Step 0:** Initializing.

- Set $R \leftarrow \emptyset$, $C \leftarrow \emptyset$, and flag = 0.

**Step 1:** Terminating.

Pick an individual $M$ from the population that has not undergone the mutation operation. If there is no such $M$, then STOP.

**Step 2:** Matching elimination.

For every edge $(i, j)$ in $M$ do:

- Generate a random variate $u$ sampled from a uniform distribution over the range $(0, 1]$.
- If $u < p_m$, then:
  - Set $M = M \setminus \{(i, j)\}$, $R = R \cup \{(i, j)\}$, $C = C \cup \{(i, j)\}$, and flag = 1.

**Step 3:** Matching reconstruction.

If flag = 0, go to Step 1. Otherwise, do:

- Set $R = R \setminus \{|V|\}$, $C = C \setminus \{1\}$, and flag = 0.

Go to the reconstruction procedure for generating new population (Section 2.5).

### 2.8. Experiments

To test the quality of the solutions produced by the proposed genetic-based algorithm, we let $|V| = 10, 20, 30, 40, \text{ and } 50$, and ten instances of the minimum WMP were randomly generated for every problem size. For each instance, the weights $w_{ij}$ were randomly sampled from a uniform distribution over the range $[100, 300]$. For each of these instances, 30 independent runs of the genetic-based algorithm were conducted. The optimal value for each of these instances was separately obtained by using Blossom IV, an implementation of the Blossom Algorithm by Rohe and Cook [9]. This code is publicly available for research purposes at http://www.or.uni-bonn.de/home/rohe/matching.html.

The parameters of the genetic-based algorithm were set as follows: population size $P = 300$; maximum number of generations $N = 600$; probability of crossover $p_c = 0.70$, and; probability of mutation $p_m = 0.20$.

| $|V|$  | Deviation | (0%, 1%) | (1%, 2%) |
|-----|-----------|----------|----------|
| 10  | 100.00    | 0.00     | 0.00     |
| 20  | 98.00     | 2.00     | 0.00     |
| 30  | 83.00     | 16.33    | 0.67     |
| 40  | 47.67     | 51.67    | 0.67     |
| 50  | 19.67     | 77.67    | 2.67     |

Table 1: 
Average deviation with respect to the optimum (in %)
Table 1 shows the deviation with respect to the optimum calculated by computing the average over 300 runs for $|V| = 10, 20, 30, 40,$ and $50.$

To analyze these results in the proper perspective, let us consider the dimension of the search space.

**Lemma 1.** Let $\mathcal{M}$ be the set of all different matchings in $G = (V,E).$ The number of total matchings is given by

$$|\mathcal{M}| = \frac{|V|!}{(|V|/2)! \cdot 2^{|V|/2}}$$

**Proof 1.** Let $C(n,m)$ and $P(n,m)$ be the number of combinations and permutations of selecting $m$ objects from $n$ objects, respectively. The first edge can be selected in $C(|V|,2)$ ways, the second one in $C(|V|−2,2)$ ways, etc. Because the order in which the edges are picked is not important, it is necessary to divide by $P(|V|/2,|V|/2).$ □

Due to the combinatorial nature of the problem, almost any feasible choice of $P$ or $P \times N$ is insignificant compared to the huge number of matchings in the search space (see Table 2). Being able to find the optimum consistently or having very small deviations with respect to it (as shown in Table 1), lead us to conclude that the proposed genetic-based algorithm can be used as a framework for solving variations of the WMP.

Because the genetic-based algorithm is merely a probabilistic search, we can derive the required number of runs to achieve optimality, given a certain level of confidence and a certain class of problems.

Let $p$ be the probability of finding the optimum in one run and $\hat{p}$ its estimate. Let $K$ be the number of trials needed to reach the optimum with a confidence level $\alpha.$ We are interested in calculating $P(\mathcal{A}),$ i.e., the probability that in $K$ trials at least once the optimum is achieved. This can be accomplished using Lemma 2.

**Lemma 2.** With at least probability $\alpha,$ the number of runs needed to reach the optimum at least once is

$$K = \left\lceil \frac{\ln(1-\alpha)}{\ln(1-p)} \right\rceil$$

**Proof 2.** It follows from $P(\mathcal{A}) = 1 - (1-p)^K \geq \alpha.$ □

With the sample of 300 runs for each $|V|,$ the estimates of $p$ (i.e., $\hat{p}$) were extracted from the column showing zero-deviation with respect to the optimum in Table 1. With the values of $\hat{p}$ and the result of Lemma 2, we constructed Table 3 to show the number of runs needed to achieve optimality with $\alpha = 0.99.$

As it is shown in Table 3, for the class of problems we generate and for the parameters fixed at $P = 300, N = 600, p_c = 0.70,$ and $p_m = 0.20,$ the required number of runs to reach optimality (with at least 99% of confidence) is 2, 3, 8 and 22, for $|V| = 20, 30, 40$ and $50,$ respectively. Therefore, the proposed algorithm requires low computational effort to reach very high quality solutions.

The average CPU time needed to obtain the best solution and the average CPU time needed to finish 600 iterations for the 300 runs (i.e., 10 problems with 30 independent seeds) conducted on a 400 MHz Pentium II machine are shown in Fig. 3. Note that in our experiments, the time needed to finish 600 iterations is almost linear in terms of the number of nodes.
2.9. Tuning parameters

There are four parameters, namely, the population size $P$, maximum number of generations $N$, probability of crossover $p_c$, and probability of mutation $p_m$, that have to be fixed in the proposed genetic-based algorithm for solving the WMP problem. Tuning parameters could help us understand the range of robustness of the proposed algorithm.

We conducted an exhaustive experiment on a set of five randomly generated problems of size $|V| = 50$. Each problem was run for a specific choice of $P$, $N$, $p_c$, and $p_m$. The weights are randomly generated from a uniform distribution over the range $[100, 500]$ for the first problem, $[5, 25]$ for the second problem, $[5, 900]$ for the third problem, and $[100, 300]$ for the fourth and fifth problems. The different levels in which the parameters were tuned is given in Table 4. The results obtained for the first problem are presented in Fig. 4.

As expected, the quality of the solution improves as the factor $P \times N$ grows. Nevertheless, it is more effective to increase $P$ than $N$.

### Table 4

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>100, 300, 500</td>
</tr>
<tr>
<td>$N$</td>
<td>200, 600, 1000</td>
</tr>
<tr>
<td>$p_c$</td>
<td>0.1, 0.2, …, 0.9</td>
</tr>
<tr>
<td>$p_m$</td>
<td>0.1, 0.2, …, 0.9</td>
</tr>
</tbody>
</table>

The conducted experiment shows that the algorithm is sensitive to the probability of crossover, $p_c$. By increasing $p_c$, the quality of the solution improves in a significative way. Moreover, the range in which the parameter can be tuned is quite broad, giving some robustness to the algorithm. The algorithm is almost insensitive to the variations of $p_m$. We believe that the strong bias towards the crossover operator is due to the fact that this operator has a mutation component in it, even though not as randomized as the mutation operator. It is worth noting that the reconstruction phase may be responsible of reverting the effect of the mutation operator.

A final remark should be noted with respect to the weights distribution. As it can be seen in the results (Figs. 4, 6–9), as the ratio between the upper and lower limit grows, there is need of more computational effort.

Finally, a choice of $P = 300$ and $N = 600$ seems to be a good tradeoff between accuracy (i.e., deviation with respect to the optimum) and computational effort, for the test problems considered (i.e., $|V| = 50$). Once again, this choice has to be seen under the light of the combinatorial nature of the problem. Roughly speaking we are exploring only $3 \times 10^{-27}$ of the search space (see Table 2). A choice of $p_c \geq 0.5$ and $p_m \approx 0.2$ seems to be reasonable.

3. Multi-criteria weighted matching

Decision makers often face the problem of choosing alternatives considering multiple conflicting criteria. For many well-studied problems, traditional optimization methods may handle very efficiently the single objective case, but fail to extend. In this section, we show that the proposed framework can be easily extended to solve the multi-criteria WMP.

### 3.1. Background

Consider a multi-criteria optimization problem with $K$ objectives over the feasible region $S$. 

![Fig. 3. Average computing time.](image-url)
Fig. 4. Tuning test problem (1): \( w_{ij} \sim U[100, 500] \) for all \( (i, j) \in E \).
\[
\begin{align*}
\text{max} & \quad \{v_1(x), v_2(x), \ldots, v_k(x)\} \\
\text{s.t.} & \quad x \in \mathcal{I} \\
\end{align*}
\]

(1)

**Definition 1.** For \( x \in \mathcal{I} \), \( v(x) \triangleq (v_1(x), v_2(x), \ldots, v_k(x)) \) is called a criterion vector, and \( \mathcal{V} = \{v(x)|x \in \mathcal{I}\} \) is called the criterion space.

**Definition 2.** A solution \( x \in \mathcal{I} \) is efficient or Pareto optimal to Problem (1) if and only if there does not exist any \( x' \in \mathcal{I} \) such that \( v_k(x') \geq v_k(x) \), for \( k = 1, \ldots, K \), and \( v_k(x') > v_k(x) \) for at least one \( k \). Otherwise, \( x \) is an inefficient solution. The set of all Pareto optimal solutions is denoted by \( \mathcal{P}^* \).

**Definition 3.** Let \( v(x), v(y) \in \mathcal{V} \) be two criterion vectors. Then, \( v(x) \) dominates \( v(y) \) if and only if \( v_k(x) \geq v_k(y) \), for \( k = 1, \ldots, K \), and \( v_k(x) > v_k(y) \) for at least one \( k \). The notation is \( v(x) \succ v(y) \).

**Definition 4.** Let \( \bar{v} \in \mathcal{V} \). Then, \( \bar{v} \) is non-dominated if and only if there does not exist any \( v \in \mathcal{V} \) that dominates \( v \). Otherwise, \( v \) is a dominated criterion vector.

**Definition 5.** The Pareto front \( \mathcal{P} \mathcal{F}^* \) for Problem (1) is defined as

\[
\mathcal{P} \mathcal{F}^* \triangleq \{v(x) \in \mathcal{V}|x \in \mathcal{P}^*\} 
\]

A good reference on multi-criteria optimization is the book by Steuer [44]. Two recent reviews have surveyed the area of evolutionary algorithms for multi-criteria optimization [8,48]. Other less recent surveys are [19,31,45]. An annotated bibliography by Ehrgott and Gandibleux [16] concentrates just on multi-objective combinatorial optimization.

As before, we assume that \( G = (V, E) \) is an undirected graph, with weights \( w_{ij}^k \geq 0 \) for \( (i,j) \in E \) and criterion \( k \). As before, the graph is assumed to be complete with an even number of nodes.

The multi-criteria weighted perfect matching problem is to find a perfect matching \( M \) of \( G \) considering all objectives simultaneously. Similar to the single objective case, we call this problem the mc-WMP:

\[
\begin{align*}
\text{max} & \quad \left\{ \sum_{(i,j) \in M} w_{ij}^1, \sum_{(i,j) \in M} w_{ij}^2, \ldots, \sum_{(i,j) \in M} w_{ij}^K \right\} \\
\end{align*}
\]

(3)

Normally the objectives in (3) conflict each other. We are interested in search the “space of tradeoffs” [44] to provide the decision makers with a set of Pareto optimal solutions.

### 3.2. Genetic-based algorithm framework

Notice that for an mc-WMP, the Pareto optimal set can be extremely large. From a decision maker’s perspective, handling such a vast amount of information becomes a problem itself. For this reason, our goal is to discover a smaller, but significant and representative sample of the Pareto optimal set, namely the approximate Pareto optimal set (also known as \( P_{\text{known}} \) [48]). It is important to note that due to the random search nature of the evolutionary algorithm, some of the elements in the approximate Pareto optimal set may not be truly Pareto optimal.

The concept of selective pressure is used to direct the genetic-based algorithm’s search toward promising regions in the search space balancing the tradeoff between exploration and exploitation [26]. The selective pressure used to obtain the approximate Pareto optimal set is designed through an adaptive weighted sum approach similar to the one used by Gen et al. [26,27] and Zhou and Gen [49]. To obtain diversity in the Pareto frontier the adaptive weighted sum approach increases the pressure on those criteria that should be further improved using information on the current approximate Pareto optimal set. By putting larger weights on those criteria with wider gaps, the algorithm exploits regions in which better non-dominated solutions are to be obtained. Notice that our idea is perfectly aligned with the suggestion of Van Veldhuisen and Lemont [48] when they comment on the possibility of using \( P_{\text{known}} \) to add diversity into the current population and to explore “holes” in the Pareto front.

In this section we outline the proposed genetic-based algorithm, which uses the framework for solving the WMP problem presented in Section 2 as the building block.
Let $N$ be the maximum number of generations, $P(t)$ the population of solutions in generation $t$, $C(t)$ the population generated through the mutation and crossover operations in generation $t$, $E(t)$ the expanded population in generation $t$, and $D$ the approximate Pareto optimal set generated by the genetic-based algorithm. The algorithm for the mc-WMP follows.

**Genetic-based Algorithm for the mc-WMP:**

1. **Initialize** $D \leftarrow \emptyset$
2. **Initialize** $P(t)$
3. **Evaluate** $P(t)$
4. **Update** $D$ with the non-dominated solutions in $P(t)$
5. **Preset selective pressure**
6. **While** $t \leq N$
7. **Recombine** and vary $P(t)$ and generate $C(t)$
8. **Evaluate** $E(t) \triangleq P(t) \cup C(t)$
9. Select $P(t + 1)$ from $P(t)$ and $C(t)$
10. **Update** the approximate Pareto optimal set $D$
11. **Update selective pressure**
12. **$t \leftarrow t + 1$**

### 3.3. Generating initial population

Let $R$ and $C$ be the sets of available rows and columns to pick from to form a new matching, respectively. Let $M$ be the matching generated by the procedure. For each criterion $k$, let $W_k$ be the weight associated with the selective pressure on the $k$th criterion (further explained in Section 3.6).

**Step 0:** Initializing.

Set $R \leftarrow \{1, 2, \ldots, |V| - 1\}$, $C \leftarrow \{2, 3, \ldots, |V|\}$, $M \leftarrow \emptyset$, $W_k \leftarrow 1/K$, $\bar{R} \leftarrow R$, and $\overline{C} \leftarrow C$.

**Step 1:** Terminating.

If $R = \emptyset$ and $C = \emptyset$, then STOP.

**Step 2:** Picking random edges.

For each $i \in R$ and $j \in C$, let $\delta_{ij} = 1$, if $i < j$; $\delta_{ij} = 0$, otherwise.

For each $i \in \bar{R}$, let the probability of selecting node $i$ be $f_i = \sum_{j \in C} \delta_{ij} / \sum_{j \in C} \sum_{i \in \bar{R}} \delta_{ij}$, if $i \in R$; $f_i = 0$, otherwise. Also define the cumulative probability $g_i = \sum_{i \in \bar{R}, j \leq i} f_r$. Set $g_0 = 0$.

Generate a random variate $u$ from a uniform distribution over the range $[0, 1]$. Pick the $r$th row (node) such that $g_{r - 1} < u \leq g_r$.

Let $C' = \{j : j > r, j \in C\}$. With the convention of $0/0 = 0$, for each edge $(\overline{r}, \overline{j})$ with $j \in C'$, define the normalized weight

$$w_{\overline{r}j} = \left( w_{\overline{r}j} / \sum_{c \in C} (w_{cj})^2 \right)^2, \text{ for } k = 1, \ldots, K,$$

and the aggregated weight considering all criteria $w_{\overline{r}j} = \sum_{k=1}^{K} w_{\overline{r}j} W_k$.

For each $j \in C$, the probability of choosing vertex $j$ given that vertex $\overline{r}$ has been chosen is given by

$$p_{\overline{r}j} = w_{\overline{r}j} / \sum_{c \in C} w_{cj}, \text{ if } j \in C; p_{\overline{r}j} = 0, \text{ otherwise.}$$

Also define $q_j = \sum_{c \in C, \overline{r}j \in C} p_{\overline{r}j}$. Set $q_{\overline{r}} = 0$.

Generate a random variate $u$ from a uniform distribution over the range $[0, 1]$. Pick the $j$th column (node) such that $q_{\overline{r} - 1} < u \leq q_{\overline{r}}$.

If $\sum_{j \in C} w_{\overline{r}j} = 0$, then pick the $j$th node randomly from $C'$.

**Step 3:** Updating.

$R \leftarrow R \setminus \{\overline{r}, j\}$, $C \leftarrow C \setminus \{\overline{r}, j\}$, and $M \leftarrow M \cup \{(i, j)\}$. Go to Step 1.

### 3.4. Evaluation

Let $M_p$ be the $p$th individual (i.e., matching) in the population. Define $x^p_{ij} = 1$, if edge $(i, j) \in M_p$; $x^p_{ij} = 0$, otherwise. Let the value for the $k$th criterion be

$$v_k(M_p) = \sum_{(i,j) \in E} w^k_{ij}.$$ 

For each individual $M_p \in E(t) \triangleq P(t) \cup C(t)$, let the normalized value of criterion $k$ be

$$\bar{v}_k(M_p) = \left( v_k(M_p) / \sqrt{\sum_{M \in E(t)} (v_k(M_p))^2} \right)^2.$$ 

Also define the fitness of the $p$th individual by $f(M_p) = \sum_{k=1}^{K} \bar{v}_k(M_p) W_k$, where $W_k$ is the weight associated with the selective pressure on the $k$th criterion ($0 \leq W_k \leq 1$).

### 3.5. Dominance

The procedure presented in this section is invoked every time when the approximate Pareto optimal set needs to be updated.

Let $v(M_p) = (v_1(M_p), v_2(M_p), \ldots, v_K(M_p))$ be the criterion vector for $M_p$. $D$ an approximate Pareto optimal set generated by the genetic-based algorithm, and $\overline{M}_d$ the $d$th individual in $D$. Also let
flag = 1, if the pth individual is efficient; flag = 0, otherwise.

Step 0: Initialization.
Let \( B = E(t) = P(t) \cup C(t) \).

Step 1: Termination.
If \( B = \emptyset \), then STOP.

Step 2: Checking Pareto optimality.
Pick any \( M_p \in B \) and do:
flag \( \leftarrow 1 \)
If \( D = \emptyset \), then go to Step 3.
For each \( M_d \in D \) do:
If \( v(M_p) > v(M_d) \), then \( D = D \setminus \{ M_d \} \).
If \( v(M_d) > v(M_p) \), then flag \( \leftarrow 0 \) and go to Step 3.

Step 3: Updating the approximate Pareto optimal set.
If flag \( = 1 \), then \( D = D \cup \{ M_p \} \) and go to Step 1.

3.6. Selective pressure

Using the same notation introduced in Section 3.5, the following procedure updates the weights of the aggregated measure used in the selective pressure mechanism.

Step 0: Initializing.
For each \( k, W_k \leftarrow 0 \).

Step 1: Normalizing.
For each \( M_d \in D \) and \( k = 1, \ldots, K \), compute
\[
\bar{v}_k(M_d) = \left( \frac{v_k(M_d)}{\sqrt{\sum_{M_d \in D} (v_k(M_d))^2}} \right)^2
\]

Step 2: Calculating the selective pressure.
For each \( k = 1, \ldots, K \), calculate
\[
W_k = \frac{\bar{v}_k^{\text{max}} - \bar{v}_k^{\text{min}}}{\sum_{k=1}^{K} (\bar{v}_k^{\text{max}} - \bar{v}_k^{\text{min}})}
\]
where,
\[
\bar{v}_k^{\text{max}} = \max_{M_d \in D} \{ \bar{v}_k(M_d) \}
\]
\[
\bar{v}_k^{\text{min}} = \min_{M_d \in D} \{ \bar{v}_k(M_d) \}
\]

3.7. Selection

Once the criterion vector is mapped into a scalar (see Section 3.4), the selection step on the genetic-based algorithm for solving the mc-WMP is exactly the same as the one presented for the WMP, namely the \((\mu + \lambda)\) selection strategy (see Section 2.6).

3.8. Genetic operators

The crossover operator for the mc-WMP is very similar to the one presented in Section 2.7.1 for the WMP. The only difference is the probability of choosing edge \( e_m \in M_s \), which is now given by \( p_{em} = \frac{\bar{w}_{em}}{\bar{w}_k(M_d)} \) and the normalized weight of the \( k \)th criterion for edge \( e_m \) is given by

\[
\bar{w}_k^{\text{max}} = \left( \frac{w_{em}}{\sqrt{\sum_{i=1}^{|M_s|} (w_{ei}^k)^2}} \right)^2
\]

The mutation operator is identical to the one presented in Section 2.7.2.

3.9. Alternative approach: multi-objective simulated annealing

Ulungu et al. [46] proposed a novel method called MOSA for solving multi-objective combinatorial optimization problems. MOSA has been successfully applied, but is not restricted, to the multi-criteria knapsack problem [46].

MOSA is based on simulated annealing [34] that originates from an analogy with the annealing process in thermodynamics and metallurgy, where the goal is to reach a minimum energy state upon cooling a substance, but slowly enough in order to avoid an undesirable final state. Simulated annealing-based algorithms like MOSA, allow non-improving moves to a neighboring solution with a probability that decreases over time based on a cooling schedule. At the beginning, almost any move in the search space is accepted, but as the algorithm progresses, it becomes more selective in accepting new solutions [28,39].
MOSA extends the principle of simulated annealing to the multi-criteria case by using scalarizing strategies [46]. Among several strategies, the most common is the criterion scalarizing strategy. In this strategy, MOSA uses an scalarizing function to aggregate the information of the multiple criteria into one scalar to calculate the improvement in one step of the algorithm. The most widely used scalarizing function is based on a weighted sum. In summary, to generate a significant and representative portion of the Pareto optimal set, MOSA requires the independent execution of \( L \) runs with weights given by \( \lambda_i = (\lambda_{i1}, \ldots, \lambda_{iK}) \) for all \( i \in \{1, \ldots, L\} \), where \( \sum_{k=1}^{K} \lambda_{ik} = 1 \) and \( \lambda_{ik} \geq 0 \).

Ideally, each of these vectors of weights emphasizes the search for non-dominated criteria vectors on a given region of the Pareto front.

We have implemented MOSA with a weighted sum scalarizing function as an alternative method to solve the mc-WMP. Let \( M_n \subseteq E \) be the current matching (solution) at iteration \( n \). A neighbor of \( M_n \) is defined as follows.

**Definition 6.** Let \( (i_1, j_1), (i_2, j_2) \in M_n \) and \( \overline{M}_n = M_n \setminus \{ (i_1, j_1), (i_2, j_2) \} \). Then \( M'_n = \overline{M}_n \cup \{ (i_1, i_2), (j_1, j_2) \} \) and \( M''_n = \overline{M}_n \cup \{ (i_1, j_2), (i_2, j_1) \} \) are called neighbors of \( M_n \).

The neighborhood of \( M_n \) has been defined in such a way that after a finite number of steps, any solution can be generated. The following lemma shows that it is possible to count (and generate) the neighbors of \( M_n \).

**Lemma 3.** Let \( M_n \) be a matching from \( G = (V, E) \) at the \( n \)th iteration of the algorithm. The number of neighbors of \( M_n \) is \( (1/4)(|V|^2 - 2|V|) \).

**Proof 3.** It follows from Definition 6. The total number of ways of selecting two edges from \( M_n \) is \( C(|V|/2, 2) \), where \( C(a, b) \) is the number of combinations of selecting \( b \) objects from \( a \) objects. In addition, each of these combinations produces two neighbors. □

MOSA has several structural parameters: the initial acceptance probability \( p_0 \), or alternatively, the initial temperature \( T_0 \); the length of the temperature step in the cooling schedule \( N_{\text{step}} \); the cooling factor \( \alpha \); and the stopping rule parameters \( N_{\text{stop}} \) and \( T_{\text{stop}} \), the maximum number of iterations without improvement and the final temperature, respectively.

To fix the structural parameters for the mc-WMP, we take a similar approach as Ulungu et al. [46]. We consider an instance of the single criterion WMP with \( |V| = 30 \) and weights \( w_j \) uniformly distributed over the range \([100, 300]\). Using Blossom IV by Rohe and Cook [9], we obtained a matching with maximal value of 4387.

To find the best set of structural parameters for MOSA, a total of 125 experiments were designed with different values of \( p_0, N_{\text{step}} \), and \( \alpha \). For each design, a total of 10 randomly independent runs were made. The parameters are set for \( p_0 = 0.6, 0.7, 0.8, 0.9, 0.95; N_{\text{step}} = 250, 500, 1000, 1500, 2000 \); and \( \alpha = 0.6, 0.7, 0.8, 0.9, 0.95 \). The stopping criteria were set to \( N_{\text{stop}} = 500 \) and \( T_{\text{stop}} = 0.001 \).

After making 1250 runs of MOSA, the best set of parameters out of the 125 designs was obtained at \( p_0 = 0.8, N_{\text{step}} = 2000 \), and \( \alpha = 0.95 \). For this design, the mean error with respect to the optimal solution is 0.2918% with a standard deviation of 0.2075%; the minimum and maximum errors in any run are 0% and 0.5471%, respectively; and the optimal value is achieved in 2 out of the 10 experiments.

### 3.10. Numerical examples

In this section, we apply the proposed genetic-based algorithm (Section 3.2) and MOSA (Section 3.9) on instances of the mc-WMP with two and three criteria in different sizes \( |V| = 10, 14, 30, \) and 50. Before presenting the results, we introduce the performance metrics used for the evaluation.

#### 3.10.1. Performance metrics

The design of performance metrics for multi-criteria combinatorial algorithms is an active area of research. Laumanns et al. [35] proposed an online performance assessment method that measures the quality of solutions dynamically as the algorithm evolves. However, most work focuses on the quality of the final approximate Pareto optimal
set generated by an algorithm [20,35,47,50–52]. In this study, we use a performance metric proposed by Zitzler and Thiele [51].

Zitzler and Thiele [51] measure the size of the dominated space (area, volume, or hypervolume) defined by the non-dominated vectors associated with the approximate Pareto optimal set obtained by a given algorithm. In other words, the larger this space is, the better the quality of the obtained approximate Pareto optimal set. Fig. 5 provides examples of the dominated space metric, where the area (Fig. 5a) and volume (Fig. 5b) shown in gray are the values of the performance measure for scenarios with two and three criteria, respectively. Every non-dominated criteria vector, marked with the symbol “•”, generates a dominated space. The hypervolume of the union of these dominated spaces by the non-dominated criteria vectors of the Pareto front is the final value of the performance metric.

This dominated space metric depends greatly on the units and magnitude on the criteria under consideration. However, for the mc-WMP it is possible to refine this performance metric by considering lower and upper bounds on the dominated space metric. In Fig. 5, the symbol “★” represents the criteria vector for the ideal solution \( \mathbf{x}^* \). The ideal solution is a vector formed by the single-criterion optimal solutions and is the best possible solution that a problem with multiple criteria may have. However, it is only achievable when there is no trade-off among the criteria. Note that the ideal solution is easy to compute in the case of the mc-WMP because the single objective WMP can be solved to optimality using the Blossom Algorithm [9]. Thus, the dominated space metric is bounded above by the hypervolume defined by the criteria vector of the ideal solution. Whereas, in the absence of a feasible solution for the mc-WMP problem, the dominated space metric is zero because there is no criteria vector defining a dominated space. Therefore, the dominated space metric is bounded below by zero.

We define the following refined metrics based on the idea of the dominated space metric:

\[
\Psi_1(\mathcal{P}_F, \mathbf{x}^*) = \frac{\eta(\mathcal{P}_F)}{\eta(\{v(\mathbf{x}^*)\})} \tag{4}
\]

and

\[
\Psi_2(\mathcal{P}_F^+, \mathcal{P}_F^*) = \frac{\eta(\mathcal{P}_F^+)}{\eta(\mathcal{P}_F^*)} \tag{5}
\]

where \( \eta \) is a function that calculates the hypervolume given a set of non-dominated criteria vectors; \( \mathcal{P}_F^+ \) and \( \mathcal{P}_F^* \) are the set of non-dominated vectors in the true and approximate Pareto front, respectively (see Definition 5); and \( \mathbf{x}^* \) and \( v(\mathbf{x}^*) \) are the ideal solution and its corresponding criteria vector, respectively.
The dominated space metrics $\Psi_1$ and $\Psi_2$ are conveniently defined so that $0 \leq \Psi_1 \leq \Psi_2 \leq 1$. Note that as these measures approach 1, the closer the approximate and the true Pareto front are to each other. In the case the true Pareto optimal set is known, $\Psi_2$ measures the error between the approximate and true Pareto fronts, whereas $\Psi_1$ is always an approximation of this error.

To evaluate the performance of the proposed algorithm and MOSA on the mc-WMP, an extensive set of experiments have been conducted. These results are organized by the number of criteria.

### 3.10.2. Bi-criteria

The problem set consists of instances for the mc-WMP with 10, 14, 30, and 50 vertices. Each instance considers two conflicting objectives. For each edge $(i,j) \in E$, $w_{ij}$ is uniformly distributed over the range $[100, 300]$ and

$$ w_{ij} = \frac{\left( \max_{(i,j) \in E} w_{ij} - \min_{(i,j) \in E} w_{ij} \right)^2}{w_{ij} + u \left( \max_{(i,j) \in E} w_{ij} - \min_{(i,j) \in E} w_{ij} \right)} $$

where $u$ is randomly selected over the range $[0, 1]$. For the smaller cases of $|V| = 10$ and 14, the true Pareto front was calculated by exhaustive enumeration.

For the genetic-based algorithm presented in Section 3.2, the parameters were chosen following the recommendations given in Section 2.9, namely, $p_c = 0.70; p_m = 0.20; P = 20, 100, 200, 250$, for $|V| = 10, 14, 30, 50$, respectively; and $N$ is set to be 500 for problems with $|V| = 10$, and 1000 for $|V| = 14, 30, 50$. For each problem size, 20 independent runs of the proposed algorithm were performed.

For MOSA, the parameters were chosen following the recommendations given in Section 3.9, namely, $p_{0} = 0.8, N_{\text{step}} = 2000, \pi = 0.95, N_{\text{stop}} = 500$, and $T_{\text{stop}} = 0.001$. The scalarizing weights ($L = 5$) were set to $\lambda_1 = (0, 1), \lambda_2 = (0.25, 0.75), \lambda_3 = (0.5, 0.5), \lambda_4 = (0.75, 0.25), \lambda_5 = (1, 0)$. For each combination of problem size and vector of scalarizing weights, 20 independent runs of MOSA were performed.

Tables 5 and 6 show the results for the dominated space metrics $\Psi_1$ and $\Psi_2$, respectively. The columns labeled “μ” and “σ” show the average and standard deviation, respectively, of the dominated space metric for 20 independent runs. The first rows in these tables (labeled M&F) correspond to the results of the dominated space metric for the algorithm presented in Section 3.2. For MOSA, we present $\Psi_1$ and $\Psi_2$ as functions of the approximate Pareto front $\mathbb{PF}^*(\lambda_i)$ generated by the algorithm using the vector of scalarizing weights $\lambda_i$. In addition, the last row shows the dominated space metric after combining (off-line) all the non-dominated vectors obtained by varying the scalarizing weights with MOSA. The combined Pareto front is denoted by $\bigcup_{i=1}^{L} \mathbb{PF}^*(\lambda_i)$.

Table 6 shows that both algorithms perform very well for $|V| = 10$ and 14, where they are
able to generate the true Pareto front. For larger instances, where the true Pareto front is unknown, the M&F algorithm presented in Section 3.2 slightly outperforms MOSA. For \( |V| = 30 \), \( \mu_{\Psi_1} = 0.823 \) and \( \mu_{\Psi_1}(U_{i=1}^{j} \underset{\lambda_i \in \mathcal{F}}{\cap} \bar{\mathcal{F}}) = 0.815 \) for M&F and MOSA, respectively; whereas for \( |V| = 50 \), \( \mu_{\Psi_1} = 0.811 \) and \( \mu_{\Psi_1}(U_{i=1}^{j} \underset{\lambda_i \in \mathcal{F}}{\cap} \bar{\mathcal{F}}) = 0.794 \) for M&F and MOSA, respectively. It is worth noting that the M&F algorithm clearly outperforms MOSA for any given vector of scalarizing weights. For \( |V| = 30 \), \( \mu_{\Psi_1} = 0.823 > \max_{i \in \{1, \ldots, L\}} \mu_{\Psi_1}(\mathcal{F} \underset{\lambda_i \in \mathcal{F}}{\cap} \bar{\mathcal{F}}) = \max\{0.747, 0.750, 0.767, 0.767, 0.714\} = 0.767 \) and for \( |V| = 50 \), \( \mu_{\Psi_1} = 0.811 > \max_{i \in \{1, \ldots, L\}} \mu_{\Psi_1}(\mathcal{F} \underset{\lambda_i \in \mathcal{F}}{\cap} \bar{\mathcal{F}}) = \max\{0.702, 0.708, 0.407, 0.713, 0.654\} = 0.713 \). This shows that the selective pressure implemented on the M&F algorithm (see Section 3.6) is able to generate a good sample of the Pareto front in one run by adapting the weights dynamically as the algorithm evolves.

By comparing the results for \( |V| = 10 \) and 14 in Tables 5 and 6, it is possible to evaluate the quality of \( \Psi_1 \) as an estimate of \( \Psi_2 \). Note that in these cases, where we know that both algorithms have generated the true Pareto front, \( \Psi_2 - \Psi_1 \approx 15\% \). If we extrapolate this result for the larger instances, the approximate Pareto fronts generated by M&F and MOSA are very likely to be very close to the true, but unknown, Pareto front.

### 3.10.3. Three criteria

This problem set consists of instances of the mc-WMP with 10, 14, 30, and 50 vertices. For each \( (i, j) \in E \), \( w_{ij}^1 \) is uniformly distributed over the range \([100, 300] \);

\[
\begin{align*}
\psi_{ij}^2 &= \frac{(\max_{(i,j) \in E} w_{ij}^1 - \min_{(i,j) \in E} w_{ij}^1)^2}{w_{ij}^1 + \sigma (\max_{(i,j) \in E} w_{ij}^1 - \min_{(i,j) \in E} w_{ij}^1)}
\end{align*}
\]

where \( \sigma \) is randomly selected over the range \([0, 1] \); and \( w_{ij}^1 \) is normally distributed with \( \mu = 50 \) and \( \sigma = 15 \). For the smaller cases of \( |V| = 10 \) and 14, the true Pareto front was calculated by exhaustive enumeration.

All the parameters and number of runs for the genetic-based algorithm and MOSA were set as described in Section 3.10.2, with the exception of the scalarizing weights. The scalarizing weights \( (L = 7) \) were set to \( \lambda_1 = (1, 0, 0), \lambda_2 = (0, 1, 0), \lambda_3 = (0, 0, 1), \lambda_4 = (0.5, 0.5, 0), \lambda_5 = (0.5, 0, 0.5), \lambda_6 = (0, 0.5, 0.5), \) and \( \lambda_7 = (0.33, 0.33, 0.34) \).

Tables 7 and 8 show the results for the dominated space metrics \( \Psi_1 \) and \( \Psi_2 \), respectively.

Table 8 shows that both algorithms perform very well for the small size problems. For \( |V| = 10 \), MOSA and M&F generate Pareto fronts that contain 100% and 99% of the true dominated space, respectively. For \( |V| = 14 \), both MOSA and M&F generate the true Pareto front. For \( |V| = 30 \), there is a virtual tie between M&F and MOSA, where \( \mu_{\Psi_1} = 0.765 \) and \( \mu_{\Psi_1}(U_{i=1}^{j} \underset{\lambda_i \in \mathcal{F}}{\cap} \bar{\mathcal{F}}) = 0.762 \),
respectively. For $|V| = 50$, MOSA seems to be slightly better than M&F, where $\mu_{\Psi_1}(\bigcup_{i=1}^{18} \overrightarrow{\Psi_i}(\lambda_i), \Psi^-) = 0.730$ and $\mu_{\Psi_1} = 0.701$, for MOSA and M&F, respectively. However, it is important to note that the M&F algorithm clearly outperforms any of the independent MOSA runs for any given vector of scalarizing weights. For $|V| = 30$, $\mu_{\Psi_1} = 0.765 > \max_{i \in \{1, \ldots, 18\}} \mu_{\Psi_i}(\overrightarrow{\Psi_i}(\lambda_i), \Psi^-)$, $\mu_{\Psi_1}(\overrightarrow{\Psi_i}(\lambda_i), \Psi^-) = \max \{0.582, 0.619, 0.644, 0.629, 0.590, 0.630, 0.671\} = 0.671$ and for $|V| = 50$, $\mu_{\Psi_1} = 0.701 > \max_{i \in \{1, \ldots, 18\}} \mu_{\Psi_i}(\overrightarrow{\Psi_i}(\lambda_i), \Psi^-) = \max \{0.494, 0.536, 0.575, 0.536, 0.506, 0.551, 0.576\} = 0.576$. This shows, that the selective pressure implemented on the M&F algorithm (see Section 3.6) is able to generate a good sample of the Pareto front in one run by adapting the weights dynamically as the algorithm evolves.

By comparing the results for $|V| = 10$ and 14 in Tables 7 and 8, it is possible to evaluate the quality of $\Psi_1$ as an estimate of $\Psi_2$. Note that in these cases, where we know that both algorithms have generated the true Pareto front, $\Psi_2 \approx 12\%$ and $\Psi_2 - \Psi_1 \approx 18\%$, for $|V| = 10$ and 14, respectively. If we extrapolate this result for the larger instances, the approximate Pareto fronts generated by M&F and MOSA are very likely to be very close to the true, but unknown, Pareto front.
4. Concluding remarks

We have presented a genetic-based framework for solving the mc-WMP. This work uses an adaptive aggregating function (weighted sum) that combines all objectives into a single individual fitness. Benefits of the adaptive aggregating function approach include that (i) it is easily implementable; (ii) it is computationally efficient [8]; (iii) it provides good empirical performance; and (iv) it extends naturally and does not need extra parameters (with respect to the single objective scenario). Although researchers have pointed out some problems, in particular, Daas and Dennis [11] discussed why a weighted sum approach does not work properly when the shape of the Pareto front is not convex, regardless of the weights used. Nevertheless, this comment should be viewed with caution in the area of combinatorial optimization, where it is not customary to have continuity and differentiability requirements such as the ones exhibited in the problem setting of [11].

We have compared the proposed genetic-based algorithm to an implementation of the MOSA [46]. Overall, both algorithms perform very well on the mc-WMP. However, the proposed genetic-based algorithm requires fewer parameters than MOSA to achieve a similar solution quality. The genetic-based algorithm requires the adjustment and control of only four parameters \((N, P, p_c, \text{ and } p_m)\), while MOSA requires user’s input of \(L(K - 1) + 5\) parameters \((p_0, N_{\text{step}}, \alpha, N_{\text{stop}}, T_{\text{stop}})\), and \(\lambda_{lk}\), for \(l = 1, \ldots, L, k = 1, \ldots, K\). For instance, in the examples presented in Sections 3.10.2 and 3.10.3, the total number of parameters required by MOSA are 10 and 19, respectively.

From a pragmatic perspective, a user of the proposed genetic-based algorithm requires only one run to generate an approximate Pareto front, thanks to the embedded adaptive weights mechanism used to update the selective pressure. On the contrary, the MOSA approach for the mc-WMP, requires the independent execution of \(L\) runs and a posteriori consolidation of the approximate Pareto front based on those runs.

Nevertheless, we were positively impressed by the performance of MOSA in the case of the mc-WMP. Once the neighborhood structure of the problem is determined, its implementation is simple and it obtains desirable results. It is definitely worthwhile to study, extend, and apply MOSA to other multi-criteria combinatorial optimization problems.

Because of the combinatorial nature of the problem, the Pareto optimal set grows exponentially fast as \(|V|\) increases. Presenting the entire Pareto optimal set to the decision maker can be an overwhelming task. The genetic-based approach is able to generate a significant portion of the Pareto optimal set. However, due to the likely NP-hardness of the problem, the solution may deteriorate as the size of the problem increases.

There are additional research paths that could be explored.

- Conduct a thorough comparative study on the performance of all classical multi-criteria evolutionary algorithms (e.g., NSGA [43], MOGA [18], or NPGA [32]) on the mc-WMP.
- Even though we have used sensitivity analysis to fine tune the parameters, there might be better approaches yet to be explored. In particular, it will be interesting to try a self-adapting algorithm that handles the parameter setting automatically [4,29].
- Finally, adaptation of the proposed algorithm to other extensions of WMP (e.g., with side constraints) could be explored.

Appendix A. Tuning parameters of the proposed algorithm

This appendix completes the results introduced in Section 2.9. Figs. 6–9 summarize the results of each test problem with combinations of parameters presented in Table 4.

Appendix B. Experiments for the WMP

This appendix presents detailed information of the results presented in Section 2.8 and summarized in Table 1 (Figs. 10–14).
Fig. 6. Tuning test problem (2): $w_{ij} \sim U[5, 25]$ for all $(i,j) \in E$. 
Fig. 7. Tuning test problem (3): $w_{ij} \sim U[5, 900]$ for all $(i, j) \in E$. 
Fig. 8. Tuning test problem (4): $w_{ij} \sim \mathcal{U}[100, 300]$ for all $(i, j) \in E$. 

Deviation w.r.t. Optimum:
- $[0 - 0.5\%]$
- $[0.5 - 1\%]$
- $[1 - 1.5\%]$
- $[1.5 - 2\%]$
- $[2 - 2.5\%]$
- $>2.5\%$
Fig. 9. Tuning test problem (5): \( w_{ij} \sim U[100, 300] \) for all \((i, j) \in E\).
Fig. 10. Deviation with respect to the optimum for $|V| = 10$. Performance of test problems (left). Overall performance (right).

Fig. 11. Deviation with respect to the optimum for $|V| = 20$. Performance of test problems (left). Overall performance (right).

Fig. 12. Deviation with respect to the optimum for $|V| = 30$. Performance of test problems (left). Overall performance (right).
Fig. 13. Deviation with respect to the optimum for $|V| = 40$. Performance of test problems (left). Overall performance (right).

Fig. 14. Deviation with respect to the optimum for $|V| = 50$. Performance of test problems (left). Overall performance (right).

References


