Supplementary Material 1:
The Two-Phase Algorithm to generate the efficient set for BOMILPs
AIChE Journal

In this supplementary material, the details of the two-phase algorithm, an illustrative example and an extensive comparison with other methods on benchmark problems are presented.

1. The Two-Phase Algorithm

The proposed algorithm consists of an initialization step and two main phases which are Phase-1 and Phase-2. In Phase-1 the supported efficient solutions are generated by using the weighted sum approach. In Phase-2, the Tchebycheff approach and two sub models are used to determine the non-supported efficient solutions. In the algorithms the following parameters are used.

- \( SE \): Supported efficient points in the objective space.
- \( NSE \): Non-supported efficient points in the objective space.
- \( LS \): Intervals to be explored in Phase-1.
- \( DLS \): Intervals to be explored in Phase-2.
- \( z_{int}, z_{end} \): Extreme efficient points.
- \( \varepsilon_1, \varepsilon_2 \): Allowable search intervals.
Initially, it is necessary to obtain extreme efficient points. To do that, an initialization phase is introduced before Phase-1 and Phase-2. The steps of the initialization phase are given in Algorithm 1.

In Algorithm 1, \texttt{WeightedSumModel} method is used to determine extreme efficient points. This function solves \( P_{\text{we}} \) model with three inputs which are \( z^* \) (lower bound), \( z^{**} \) (upper bound), \( \lambda \) (weights) and returns an efficient solution.

After the initialization phase, the supported efficient solutions can be obtained by using Algorithm 2.

In Phase-1, starting from the extreme efficient points interval all intervals in the \( LS \) list is investigated. For each interval \([z^l, z^u]\), \texttt{WeightedSumModel} is used to obtain supported efficient solution. There may not be any supported efficient solution for some search intervals, in that case the method returns either \( z^l \) or \( z^u \). The algorithm is terminated when the search interval list is empty. Phase-1 of the proposed method is followed by the Phase-2. In Phase-2, all non-supported efficient solutions are determined. The steps of the Phase-2 are given in Algorithm 3.
Algorithm 2: Phase-1

input : $z_{int}, z_{end}, LS$
output: $SE, DLS$

1 while $LS \neq \emptyset$ do
2     Pick an interval $INT$ from the $LS$, $INT \leftarrow [z^l \ z^u]$  
3     $LS \leftarrow LS \setminus INT$
4     $DLS \leftarrow DLS \cup [z^l \ z^u]$
5     Calculate $z^*$ and $z^{**}$ from Equation (??) and (??), respectively
6     Calculate $\lambda_1$ and $\lambda_2$ from Equation (??) and (??), respectively
7     $z^{PWe} \leftarrow \text{weightedSumModel}(z^*, z^{**}, \lambda)$
8     if ($z^{PWe} \neq z^l$) and ($z^{PWe} \neq z^u$) then
9         $SE \leftarrow SE \cup z^{PWe}$
10        $LS \leftarrow LS \cup \{[z^l \ z^{PWe}],[z^{PWe} z^u]\}$
11        $DLS \leftarrow DLS \setminus [z^l \ z^u]$
12     end
13 end
14 return $SE, DLS$

In this phase of the algorithm, non-supported efficient solutions are determined by using the $P_{Tch}$, $P_{Ep1}$ and $P_{Ep2}$ models. Similar to Phase-1, all search intervals in the $DLS$ list is explored. For each interval $[z^l \ z^u]$, $\text{TchebycheffModel}$ and two submodel methods that corresponds to $PEpModel1$ and $PEpModel2$ are used to obtain non-supported efficient solutions. The $\text{TchebycheffModel}$ function gets $z^*$ (lower bound), $z^{**}$ (upper bound) as inputs and returns $\alpha^*$. Then, functions $PEpModel1$ and $PEpModel2$ are used to obtain $z^{Pep1}$ and $z^{Pep2}$, respectively. Four different cases are controlled to determine the efficient solution(s). Phase-2 is terminated when the $DLS$ list is empty.

2. Illustrative Example

In this subsection, the proposed two-phase algorithm is applied to bicriteria knapsack problem to show how the method generates supported and non-supported efficient solutions. The bicriteria knapsack problem consists of an integer capacity $W > 0$ with $n$ objects. Each object $j$ has a positive integer weight $w_j$ and two non-negative integer profits which are $v^1_j$ and $v^2_j$. Decision variables ($x_j$) represents whether the object $j$ is selected or not. The mathematical model of the bicriteria knapsack problem is as follows;
Algorithm 3: Phase-2

input : $DLS, \varepsilon_1, \varepsilon_2$
output: $NSE$

1. Initialize $\alpha^0$
2. while ($DLS \neq \emptyset$) do
3. Pick an interval $INT$ from the $DLS$, $INT \leftarrow [z^l, z^u]$
4. $DLS \leftarrow DLS \setminus INT$
5. if $((z_2^u - z_1^l) > \varepsilon_1)$ or $((z_2^l - z_1^u) > \varepsilon_2)$ then
6. Calculate $z^*$ and $z^{**}$ from Equation (??) and (??), respectively
7. $\alpha^* \leftarrow$ TchebycheffModel ($z^*, z^{**}$)
8. if ($\alpha^* < \alpha^0$) then
9. $z^{PEp1} \leftarrow PEpModel1 (z^*, z^{**}, \alpha^*)$
10. $z^{PEp2} \leftarrow PEpModel2 (z^*, z^{**}, \alpha^*)$
11. if ($z^{PEp1} \neq z^{PEp2}$) then
12. // Solutions are equal
13. $DLS \leftarrow DLS \cup \{[z^l_z^{PEp1}, [z^{PEp1} z^u]\}$
14. $NSE \leftarrow NSE \cup z^{PEp1}$
15. end
16. if ($z^{PEp2} \Delta z^{PEp1}$) then
17. // $z^{PEp2}$ dominates $z^{PEp1}$
18. $DLS \leftarrow DLS \cup \{[z^l_z^{PEp2}, [z^{PEp2} z^u]\}$
19. $NSE \leftarrow NSE \cup z^{PEp2}$
20. end
21. if ($z^{PEp1} \Delta z^{PEp2}$) then
22. // $z^{PEp1}$ dominates $z^{PEp2}$
23. $DLS \leftarrow DLS \cup \{[z^l_z^{PEp1}, [z^{PEp1} z^u]\}$
24. $NSE \leftarrow NSE \cup z^{PEp1}$
25. end
26. if ($\sim (z^{PEp1} \Delta z^{PEp2}) \text{ and } \sim (z^{PEp2} \Delta z^{PEp1})$) then
27. // Both solutions are efficient
28. $DLS \leftarrow DLS \cup \{[z^l_z^{PEp1}, [z^{PEp1} z^u]\}$
29. $NSE \leftarrow NSE \cup z^{PEp1}$
30. $NSE \leftarrow NSE \cup z^{PEp2}$
31. end
32. end
33. return $NSE$
Equation (1) is the objective function of the model that maximizes both first and second objective functions. Equation (2) is the capacity constraint, the total weight of selected objects has to be less than or equal to the knapsack’s capacity. Equation (3) represents of binary integrality constraints.

An instance of the bicriteria knapsack problem is given below where the number items is $n = 10$ and $v_1, v_2$ and $w$ parameters are generated uniformly distributed in the range of $[1, 100]$. The knapsack capacity is calculated by using Equation (4).

$$W = \left\lfloor \frac{\sum_{j=1}^{n} w_j}{2} \right\rfloor$$

The bicriteria knapsack problems instance is given below where the parameters of the instance are obtained as explained above.

$$\max f_1(x) = 42x_1 + 47x_2 + 46x_3 + 37x_4 + 16x_5 + 54x_6 + 94x_7 + 37x_8 + 76x_9 + 59x_{10}$$
$$\max f_2(x) = 90x_1 + 75x_2 + 78x_3 + 31x_4 + 80x_5 + 23x_6 + 11x_7 + 6x_8 + 97x_9 + 16x_{10}$$

s.t.

$$14x_1 + 80x_2 + 73x_3 + 89x_4 + 66x_5 + 48x_6 + 53x_7 + 4x_8 + 27x_9 + 80x_{10} \leq 267$$

$$x_j \in \{0, 1\}, \quad j = 1, \ldots, 10$$

The proposed two-phase algorithm is a generic method and the criterion functions in the algorithm is defined in minimization form. As the bicriteria knapsack problem’s criteria functions are in maximization form, we first transformed it into minimization form ($f'_1(x) = -f_1(x)$ and $f'_2(x) = -f_2(x)$) then generate the efficient set for this instance.

For the given instance, the extreme efficient points are determined which are $z^{\text{int}} = [-362 \ - 243]$ and $z^{\text{end}} = [-264 \ - 426]$. A new supported efficient point $[-342 \ - 357]$ is obtained with convex combination of extreme efficient solutions. A final efficient point is
Figure 1: Supported and non-supported efficient solutions for the illustrative example.
which is obtained by using weighted sum model where $z^l = [-362 \ -243]$ and $z^u = [-342 \ -357]$. The supported efficient points are shown in Figure S1.

After the determination of supported efficient solutions, the two-phase algorithm continues to generate the efficient set by obtaining non-supported efficient solutions. First non-supported efficient point $[-302 \ -369]$ is determined when $z^l = [-342 \ -357]$ and $z^u = [-264 \ -426]$. Other non-supported efficient points are as follows $[-271 \ -374]$, $[-272 \ -371]$, $[-311 \ -362]$, $[-312 \ -359]$, $[-349 \ -305]$, $[-354 \ -298]$, $[-350 \ -302]$. Non-supported efficient points are also shown in Figure S1.

3. Scalarization Methods for Bicriteria Optimization

Several methods have been proposed in the literature to obtain the efficient set for biobjective optimization problems:

- The weighted-sum method (Geoffrion, 1968)
- The $\epsilon$-Constraint method (Haimes et al., 1971)
- Augmented $\epsilon$-Constraint method (Mavrotas, 2009)
- The weighted Tchebycheff method (Bowman, 1976)
- Thw augmented-Weighted Tchebycheff method (Steuer and Choo, 1983)
- Quadratic function method (Neumayer and Schweigert, 1994)
- Two-stage sub problems (Neumayer and Schweigert, 1994)

A brief description of each of these methods are given below.

3.1 The Weighted-sum Method

- Weighted sum model is defined as follows,

$$\textbf{P}_{\text{We}} \min \sum_{j=1}^{p} \lambda_j f_j(x) \quad (5)$$

s.t. $x \in \mathcal{X}$

$$\sum_{j=1}^{p} \lambda_j = 1 \quad (7)$$

$$0 \leq \lambda_j \leq 1 \quad j = 1, \ldots, p \quad (8)$$
• Optimal solution of the $P_{We}$ is efficient (Geoffrion, 1968).

• Only supported efficient solution can be obtained the weighted-sum method.

### 3.2 The $\epsilon$-constraint Method

• The $\epsilon$—constraint was proposed by (Haines et al., 1971).

• It is based on scalarization where one of the objective functions is minimized while other objective functions are bounded from above with additional constraints.

• It is one of the most widely used methods for generating the efficient set. However, it generates only the supported efficient solutions.

$$ P_{\epsilon} \quad \text{min} \quad f_k(x) $$

\hspace{1cm} \text{s.t.} \quad f_j(x) \leq \epsilon_j \quad j = 1, \ldots, p; \ j \neq k \quad (10)

\hspace{1cm} x \in \mathcal{X} \quad (11)

**Theorem 1.** For any $\epsilon_k \in \mathbb{R}^{p-1}$ the following statements hold.

• If $x^* \in \mathcal{X}$ is optimal solution of $P_{\epsilon}$, then $x^*$ is weakly efficient solution.

• If $x^* \in \mathcal{X}$ is unique optimal solution of $P_{\epsilon}$, then $x^*$ is efficient solution.

$$ P_{\text{lex} - \epsilon} \quad \text{Lex min} \quad (f_1(x), f_2(x)) \quad (12) $$

\hspace{1cm} \text{s.t.} \quad f_j(x) \leq \epsilon_j \quad j = 1, 2 \quad (13)

\hspace{1cm} x \in \mathcal{X} \quad (14)

The optimal solution $(x^* \in \mathcal{X})$ of $P_{\text{lex} - \epsilon}$ is efficient.

### 3.3 The Augmented $\epsilon$-constraint Method

• The objective function of the $\epsilon$—constraint method is turned into augmented form by (Mavrotas, 2009).
• For \( \rho > 0 \) (sufficiently small, e.g. \( \rho = 0.0001 \)) the augmented \( \epsilon \)-constraint model is defined as follows.

\[
P_{Aug-\epsilon} \quad \min \quad f_k(x) + \rho \sum_{j=1,\ldots,p; j \neq k} s_j
\]

\[
\text{s.t.} \quad f_j(x) + s_j = \epsilon_j \quad j = 1,\ldots,p; j \neq k \quad (16)
\]

\[
s_j \in \mathbb{R}^+ \quad (17)
\]

\[
x \in \mathcal{X} \quad (18)
\]

The optimal solution \( (x^* \in \mathcal{X}) \) of \( P_{Aug-\epsilon} \) is efficient.

3.4 The Weighted Tchebycheff Method

• The weighted Tchebycheff problem was introduced by (Bowman, 1976).

• The main idea of the method is to find a solution as close as possible to ideal point.

• For the weight vector \( w \geq 0 \) the weighted Tchebycheff model is defined as follows.

\[
P_{Tch} \quad \min \max \quad (w_j|f_j(x) - f^{I}_j|) \quad i = 1,\ldots,p
\]

\[
\text{s.t.} \quad x \in \mathcal{X} \quad (20)
\]

• The problem can be solved by adding \( p \) variables and constraints to linearize the max term of the objective function.

• The solution of the problem is weakly efficient.

3.5 The Augmented-weighted Tchebycheff Method

• The augmented weighted Tchebycheff was suggested by (Steuer and Choo, 1983).

• In this case, distance between ideal point and feasible objective region is minimized.

• For the weight vector \( w \geq 0 \) and \( \rho > 0 \) (sufficiently small, e.g. \( \rho = 0.0001 \)) the augmented weighted Tchebycheff model is defined as follows.
The solution of the problem is efficient.

3.6 Quadratic Function Method

• The main of this method is to hyperbola instead of norm.
• The input data are two efficient points \( p, q \in \mathbb{Z}^2 \).
• Using these efficient points parameters \( a, b \) and \( c \) are determined as follows,

\[
a = (p_1 - q_1 + \frac{1}{2})(q_2 - p_2 + \frac{1}{2}) - \frac{1}{4} \tag{23}
\]
\[
b = (p_2 - q_2 - \frac{1}{2})(p_1 - q_1 + \frac{1}{2})q_1 + \frac{1}{4}p_1 \tag{24}
\]
\[
c = (p_2 - q_2 - \frac{1}{2})(p_1 - q_1 + \frac{1}{2})p_2 + \frac{1}{4}q_2 \tag{25}
\]

\[
\text{P}_{\text{quad}} \quad \min \quad af_1(x)f_2(x) + bf_2(x) + cf_1(x) \tag{26}
\]
\[
\text{s.t.} \quad p_1 \leq f_1(x) \leq q_1 \tag{27}
\]
\[
p_2 \leq f_2(x) \leq q_2 \tag{28}
\]
\[
x \in \mathcal{X} \tag{29}
\]

• The optimal solution of \( P_{\text{quad}} \) is an efficient solution Neumayer and Schweigert (1994).
• In some cases objective function of the \( P_{\text{quad}} \) may be non-convex.

3.7 Max Ordering

• This approach is based on a two step procedure based on max ordering Neumayer and Schweigert (1994).

For a weight vector \( w \geq 0 \), the subproblem \( P_w \) is as follows;

\[
P_w \quad \min \max_{j=1,...,p} \quad w_j f_j(x) \tag{30}
\]
\[
\text{s.t.} \quad x \in \mathcal{X} \tag{31}
\]
Let optimal solution of $P_w$ as $z^*$, then the second subproblem $Q_w$ defined as follows,

$$Q_w \min \sum_{j=1}^{p} f_j(x)$$  \hspace{1cm} (32)

s.t. \hspace{0.5cm} w_j f_j(x) \leq z^* \hspace{0.5cm} j = 1, \ldots, p \hspace{1cm} (33)

$$x \in \mathcal{X}$$ \hspace{1cm} (34)

The optimal solution of $P_w$ and $Q_w$ is an efficient solution.

4. Computational Results

In this section, we provide comparative performance analysis of the proposed two-stage algorithm with the other algorithms available in the literature. We select the most widely used benchmark problems for testing the effectiveness of different algorithms. All experiments presented here were performed on a dual core Xeon 3.33 GHz CPU with 32 Gb RAM. Algorithms are coded in C++ with the IBM CPLEX 12.1 callable library.

4.1 The Assignment Problem

- The bicriteria assignment problem consists of two non-negative integer assignment costs which are $c_{ij}^1$ and $c_{ij}^2$.

- $x_{ij} = \begin{cases} 
1 & \text{if } i \text{ is assigned to } j \\
0 & \text{otherwise} 
\end{cases}$

- The mathematical model of the bicriteria assignment problem is as follows;

$$P_{Ap} \min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^1 x_{ij}$$  \hspace{1cm} (35)

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij}^2 x_{ij}$$  \hspace{1cm} (36)

s.t. \hspace{0.5cm} \sum_{j=1}^{n} x_{ij} = 1 \hspace{0.5cm} i = 1, \ldots, n \hspace{1cm} (37)

\sum_{i=1}^{n} x_{ij} = 1 \hspace{0.5cm} j = 1, \ldots, n \hspace{1cm} (38)

$$x_{ij} \in \{0, 1\} \hspace{0.5cm} \forall (i, j)$$ \hspace{1cm} (39)

- The benchmark problems for the bicriteria assignment problem instances taken from Przybylski et al. (2008).

- For sizes $n = 10$ to $n = 100$ with an increment of 10, 10 instances each.
The objective function coefficients were generated in the range of \([0, 20]\), \([0, 40]\), \([0, 60]\).

The results are shown in three tables.

In the tables, \(n\) is the number of tasks to be assigned and \(|\mathcal{X}_{SE}|\) corresponds to the number of supported efficient solutions.

Table 1: Average CPU times (sec) of tested algorithms on the assignment problem ([0, 20] range).

| \(n\) | \(|\mathcal{X}_{SE}|\) | \(P_{mAug-\epsilon}\) | \(P_{Aug}\) | \(P_{Tch}\) | Max Ord. | Two-Phase |
|------|----------------|----------------|----------------|----------------|---------|-----------|
| 10   | 29             | 0.13            | 1.42            | 3.97            | 4.09    | 0.80      |
| 20   | 49             | 0.53            | 7.21            | 18.61           | 20.81   | 5.90      |
| 30   | 98             | 1.85            | 31.72           | 82.23           | 92.64   | 28.27     |
| 40   | 106            | 2.51            | 62.66           | 153.52          | 181.61  | 56.96     |
| 50   | 117            | 3.57            | 120.21          | 285.50          | 338.27  | 110.28    |
| 60   | 181            | 9.20            | 337.56          | 737.49          | 938.64  | 355.32    |
| 70   | 176            | 14.63           | 527.67          | 1092.71         | 1243.09 | 561.35    |
| 80   | 190            | 26.01           | 851.61          | 1667.01         | 1927.94 | 925.66    |
| 90   | 216            | 56.52           | 1514.89         | 2782.79         | 3130.79 | 1702.12   |
| 100  | 230            | 277.97          | 2548.71         | 4131.25         | 4706.44 | 3185.88   |

Table 2: Average CPU times (sec) of tested algorithms on the assignment problem ([0, 40] range).

| \(n\) | \(|\mathcal{X}_{SE}|\) | \(P_{mAug-\epsilon}\) | \(P_{Aug}\) | \(P_{Tch}\) | Max Ord. | Two-Phase |
|------|----------------|----------------|----------------|----------------|---------|-----------|
| 10   | 21             | 0.10            | 0.99            | 2.53            | 2.42    | 0.56      |
| 20   | 66             | 0.63            | 8.91            | 24.17           | 25.11   | 7.30      |
| 30   | 109            | 2.11            | 33.45           | 84.94           | 96.74   | 29.82     |
| 40   | 186            | 5.07            | 114.78          | 272.71          | 347.77  | 104.34    |
| 50   | 216            | 8.77            | 234.98          | 536.95          | 690.18  | 215.57    |
| 60   | 253            | 14.64           | 468.61          | 1006.66         | 1330.05 | 493.27    |
| 70   | 331            | 22.37           | 1010.31         | 2178.18         | 2633.49 | 1074.79   |
| 80   | 355            | 33.98           | 1581.67         | 3284.27         | 3835.96 | 1719.20   |
| 90   | 432            | 55.35           | 2854.61         | 5418.72         | 6662.81 | 3207.42   |
| 100  | 429            | 72.38           | 3884.95         | 7158.59         | 8687.52 | 4856.18   |

The results indicate that \(P_{mAug-\epsilon}\) is the fastest algorithm. However, \(P_{mAug-\epsilon}\) is a single stage scalarization method and does not guarantee that all identified solutions
Table 3: Average CPU times (sec) of tested algorithms on the assignment problem ([0, 60] range).

| n  | |X_{SE}| | P_{mAug-\epsilon} | P_{Aug} | P_{Tch} | Max Ord. | Two-Phase |
|----|----------|-------------|-------------|---------|---------|----------|-----------|
| 10 | 29       | 0.13        | 1.42        | 3.97    | 4.09    | 0.80     |
| 20 | 49       | 0.53        | 7.21        | 18.61   | 20.81   | 5.90     |
| 30 | 98       | 1.85        | 31.72       | 82.23   | 92.64   | 28.27    |
| 40 | 106      | 2.51        | 62.66       | 153.52  | 181.61  | 56.96    |
| 50 | 117      | 3.57        | 120.21      | 285.50  | 338.27  | 110.28   |
| 60 | 181      | 9.20        | 337.56      | 737.49  | 938.64  | 355.32   |
| 70 | 176      | 14.63       | 527.67      | 1092.71 | 1243.09 | 561.35   |
| 80 | 190      | 26.01       | 851.61      | 1667.01 | 1927.94 | 925.66   |
| 90 | 216      | 56.52       | 1514.89     | 2782.79 | 3130.79 | 1702.12  |
| 100| 230      | 277.97      | 2548.71     | 4131.25 | 4706.44 | 3185.88  |

are efficient. Besides, \( P_{mAug-\epsilon} \) may miss some of the efficient solutions depending on the value of the penalty parameter, \( \rho \).

- The performance of \( P_{Aug} \) and the Two-Phase algorithms are clearly better than the performance of \( P_{Tch} \) and Max Ordering algorithms for all instances.
- The \( P_{Aug} \) has a slightly better performance than the Two-Phase algorithms on about half of the instances.

4.2 The Knapsack Problem

- The bicriteria knapsack problem consists of an integer capacity \( W > 0 \) with \( n \) objects.
- Each object \( j \) has a positive integer weight \( w_j \) and two non-negative integer profits which are \( v_{1j} \) and \( v_{2j} \).
- \( x_j = \begin{cases} 1 & \text{If the object } j \text{ is selected.} \\ 0 & \text{otherwise} \end{cases} \)
- The mathematical model of the bicriteria knapsack problem is as follows:

\[
P_{Kp} \quad \max \sum_{j=1}^{n} v_{1j}^1 \cdot x_j \quad (40) \\
\max \sum_{j=1}^{n} v_{2j}^2 \cdot x_j \quad (41) \\
s.t. \sum_{j=1}^{n} w_j x_j \leq W \quad (42) \\
x_j \in \{0, 1\} \quad j = 1, \ldots, n \quad (43)
\]
Table 4: Average CPU times (sec) of tested algorithms on the knapsack problem.

| n  | |X_{SE}| |X_E| P_{We} | P_{lex-ϵ} | P_{Tch} |
|----|---------|---------|--------|--------|----------|---------|
| 100| 18.9    | 159.3   | 0.19   | 3.74   | 32.90    |
| 200| 34.6    | 529.0   | 0.64   | 35.82  | 245.78   |
| 300| 54.3    | 1130.7  | 1.84   | 152.50 | 850.51   |
| 400| 68.4    | 1713.3  | 3.01   | 346.17 | 1682.42  |
| 500| 85.1    | 2537.5  | 4.77   | 682.73 | 2960.41  |
| 600| 101.6   | 3593.9  | 7.28   | 1280.96| 5086.33  |

Table 5: Average CPU times (sec) of tested algorithms on the knapsack problem (continued).

<table>
<thead>
<tr>
<th>n</th>
<th>P_{Aug}</th>
<th>Max Ord.</th>
<th>Two-Phase</th>
<th>P_{mAug-ϵ}</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>19.41</td>
<td>16.03</td>
<td>16.87</td>
<td>2.26</td>
</tr>
<tr>
<td>200</td>
<td>159.32</td>
<td>167.81</td>
<td>152.49</td>
<td>22.10</td>
</tr>
<tr>
<td>300</td>
<td>554.50</td>
<td>431.93</td>
<td>638.53</td>
<td>95.93</td>
</tr>
<tr>
<td>400</td>
<td>1111.65</td>
<td>1084.34</td>
<td>1517.71</td>
<td>215.48</td>
</tr>
<tr>
<td>500</td>
<td>1911.57</td>
<td>1986.26</td>
<td>2002.35</td>
<td>425.10</td>
</tr>
<tr>
<td>600</td>
<td>3149.32</td>
<td>3212.41</td>
<td>3105.25</td>
<td>809.71</td>
</tr>
</tbody>
</table>

- \( v_1^k \in U[1, 1000], v_2^k \in U[1, 1000] \) and \( w^k \in U[1, 1000] \).

\[
W = \left\lfloor \frac{\sum_{j=1}^{n} w_j}{2} \right\rfloor \tag{44}
\]

- Number of items varies 100 through 600 (\( n = 100, 200, \ldots, 600 \)), and 10 instances are solved for each size.
- Average of 10 instances are reported with respect to number of items
- In the tables, \( n \) is the number of items, \( |X_{SE}| \) corresponds to the number of supported efficient solutions and \( |X_E| \) is the number of efficient solutions.

- The results indicate that \( P_{We}, P_{lex-ϵ} \) and \( P_{mAug-ϵ} \) are faster than the other algorithms. This expected from single stage algorithms since neither they cannot guarantee that the solutions generated are efficient nor they can generate all efficient solutions for problems with non-convex feasible regions.
- The performance of the Two-Phase is significantly better than \( P_{Tch} \) and slightly better than \( P_{Aug} \) and Max Ordering algorithms for all instances.
4.3 Set Covering Problem

- The bicriteria set covering problem consists of a binary parameter $a_{ij}$ whether item $i$ can be covered by set $j$.
- Each set $j$ has two non-negative integer profits which are $c_j^1$ and $c_j^2$.
- $x_j = \begin{cases} 1 & \text{If the set } j \text{ is selected.} \\ 0 & \text{otherwise} \end{cases}$
- The mathematical model of the bicriteria set covering problem is as follows;

$$\begin{align*}
\text{P}_{\text{Scp}} & \quad \min \sum_{j=1}^{n} c_j^1 \ast x_j \\
& \quad \min \sum_{j=1}^{n} c_j^2 \ast x_j \\
& \quad \text{s.t.} \sum_{j=1}^{n} a_{ij} x_j \geq 1 \quad i = 1, \ldots, m \\
& \quad x_j \in \{0, 1\} \quad j = 1, \ldots, n
\end{align*}$$

- Number of items varies 100 through 600 ($n = 100, 200, \ldots, 600$), and 10 instances are solved for each size.
- Average of 10 instances are reported with respect to number of items
- In the tables, $m$ is the number of sets, $n$ is the number of elements, $|X_{SE}|$ corresponds to the number of supported efficient solutions and $|X_E|$ is the number of efficient solutions.

Table 6: Average CPU times (sec) of tested algorithms on the set covering problems.

| $m$ | $n$ | $|X_{SE}|$ | $|X_E|$ | $P_{m_{Aug-\epsilon}}$ | $P_{Aug}$ |
|-----|-----|-----------|--------|------------------|----------|
| 10  | 100 | 11        | 39     | 0.17             | 1.61     |
| 40  | 200 | 23        | 107    | 0.89             | 9.04     |
| 40  | 400 | 34        | 208    | 1.95             | 26.61    |
| 60  | 200 | 14        | 46     | 2.08             | 8.54     |
| 60  | 600 | 32        | 257    | 4.31             | 55.15    |
| 80  | 600 | 13        | 98     | 24.36            | 80.23    |
| 80  | 800 | 38        | 424    | 6.56             | 112.06   |
| 100 | 800 | 23        | 132    | 52.54            | 179.45   |
| 100 | 1000| 24        | 157    | 118.13           | 472.01   |
| 100 | 1000| 21        | 83     | 31.33            | 126.57   |
| 200 | 1000| 24        | 274    | 4631.39          | 32055.70 |
Table 7: Average CPU times (sec) of tested algorithms on the set covering problems (continued).

<table>
<thead>
<tr>
<th>m</th>
<th>n</th>
<th>Two-Phase</th>
<th>$P_{Tch}$</th>
<th>$P_{lex-\epsilon}$</th>
<th>Max Ord.</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>100</td>
<td>1.03</td>
<td>3.26</td>
<td>0.30</td>
<td>3.54</td>
</tr>
<tr>
<td>40</td>
<td>200</td>
<td>8.37</td>
<td>17.01</td>
<td>1.52</td>
<td>16.70</td>
</tr>
<tr>
<td>40</td>
<td>400</td>
<td>27.89</td>
<td>51.24</td>
<td>4.53</td>
<td>58.61</td>
</tr>
<tr>
<td>40</td>
<td>200</td>
<td>4.83</td>
<td>16.90</td>
<td>2.10</td>
<td>17.71</td>
</tr>
<tr>
<td>60</td>
<td>600</td>
<td>58.97</td>
<td>107.10</td>
<td>10.21</td>
<td>121.02</td>
</tr>
<tr>
<td>60</td>
<td>600</td>
<td>47.51</td>
<td>171.11</td>
<td>25.24</td>
<td>177.68</td>
</tr>
<tr>
<td>80</td>
<td>800</td>
<td>143.63</td>
<td>241.01</td>
<td>18.73</td>
<td>291.07</td>
</tr>
<tr>
<td>80</td>
<td>800</td>
<td>110.20</td>
<td>367.09</td>
<td>56.60</td>
<td>372.76</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>408.21</td>
<td>934.13</td>
<td>155.94</td>
<td>946.32</td>
</tr>
<tr>
<td>100</td>
<td>1000</td>
<td>73.43</td>
<td>248.54</td>
<td>34.45</td>
<td>262.06</td>
</tr>
<tr>
<td>200</td>
<td>1000</td>
<td>44213.60</td>
<td>55034.50</td>
<td>8579.11</td>
<td>54418.70</td>
</tr>
</tbody>
</table>

- The performance of the Two-Phase is significantly better than $P_{Tch}$ and Max Ordering algorithms. $P_{Aug}$ and Two-Phase have similar performance, in half of the instances the Two-Phase has a better performance than $P_{Aug}$.

The comparative analysis of different algorithms show that the Two-Phase algorithm consistently generates the efficient set faster across all of the benchmark problem categories. While some algorithms perform very well in one problem type, they show very poor performance with another type. However, the Two-Phase is consistently the fastest or tied for the fastest CPU time.

**References**


Steuer, R. E., Choo, E. U., 1983. An interactive weighted Tchebycheff procedure for multiple 
objective programming. Mathematical Programming 26 (3), 326–344.