



# A Basic Variable Neighborhood Search Approach for the Bi-objective Multi-row Equal Facility Layout Problem

Nicolás R. Uribe , Alberto Herrán , and J. Manuel Colmenar  

Department of Computer Science and Statistics, Rey Juan Carlos University,  
Mostoles, Spain

{nicolas.rodriguez,alberto.herran,josemanuel.colmenar}@urjc.es

**Abstract.** The Bi-Objective Multi-Row Facility Layout Problem is a problem belonging to the family of Facility Layout Problems. This problem is challenging for exact and metaheuristics approaches. We use the Pareto front approach instead of the weight approach by means of a non-dominated solution set which we update in order to keep only the non-dominated solutions. To tackle this problem, we propose a Basic VNS algorithm based on a constructive method that generates random solutions, a mono-objective local search that relies on an interchange move, and a shake method that applies insert moves. In this regard, we also explain how to adapt the mono-objective schema of the BVNS for a multi-objective one. Then, we compare our results with the state of the art and propose future work.

**Keywords:** Basic VNS · Bi-Objective optimization · Facility Location

## 1 Introduction

Facility Layout Problems (FLPs) are a well-known family of optimization problems with the goal of finding the optimal position of facilities in a given layout. See [1, 9] and [10] for recent surveys. FLP is seen as a challenge for both exact and heuristic procedures. The very first work addressing this family of problems originally dates from 1969 [16] and was motivated by the need for a linear arrangement of different rooms along a corridor. This problem is called Single-Row Facility Layout Problem (SRFLP).

Many FLP variants can also be found in the literature. For example, in the Single-Row Equal Facility Layout Problem (SREFLP) [11] all the facilities have the same width. Using two rows, we find the Double-Row Facility Layout Problem (DRFLP) [4], and its variants. Other variants consider several rows (more than two rows) such as the Multi-Row Facility Layout Problem (MRFLP) [12],

---

This work has been partially supported by the Spanish Ministerio de Ciencia e Innovación (MCIN/AEI/10.13039/501100011033/FEDER, UE) under grant ref. PID2021-126605NB-I00.

and its variants. Another challenging variant is the Multi-Row Equal Facility Layout Problem (MREFLP) [2, 18]. From the multi-objective point of view, the bi-objective MREFLP is a variant of the MREFLP which considers two objective functions and where both, the number of rows and the number of facilities that can be allocated in each row, are given by the instance. In other words, the layout configuration is fixed by the target instance.

In general multi-objective problems can be addressed from two different approaches. The first one combines all objective functions into a single one by means of a weighted sum, and returns a single value [3, 6–8, 13, 15], and [17]. The second approach considers a set of different non-dominated solutions taking into account the objective functions separately [17]. In this work, we will follow this last approach.

The rest of the paper is structured as follows. In Sect. 2 we provide a description for the problem and an example of evaluation. Then, we describe our Basic VNS approach in Sect. 3. In Sect. 4 we explain our results and compare them with the state of the art. Finally, in Sect. 5, we present our conclusions and future work.

## 2 Problem Description

Given a set of facilities ( $F$ ), a weight matrix corresponding to an objective function ( $W$ ), a solution that we are going to evaluate ( $\varphi$ ), the number of rows ( $m$ ) and the number of columns ( $c$ ), we can calculate the objective function value through Eq. 1 in this way: let  $\rho(i, j)$  be the facility in row  $i$  and column  $j$  in  $\varphi$ ,  $w_{u,v}$  be the weight between facilities  $u$  and  $v$  in matrix  $W$ , and  $d_{u,v}$  the distance between facilities  $u$  and  $v$ , the equation computes the sum of the products of all facilities pairwise weight and their pairwise distance.

$$\mathcal{F}(F, W, \varphi) = \sum_{i=1}^m \sum_{j=1}^c \sum_{k=1}^m \sum_{l=j+1}^c w_{\rho(i,j), \rho(k,l)} \cdot d_{\rho(i,j), \rho(k,l)} \quad (1)$$

As seen, the product has two parameters: a weight between the pairwise and the distance between these facilities. We extract the weight from the weight matrix, but we need to calculate the distance among them. In this problem, the Manhattan distance is used. This means that we will consider the vertical distance in addition to the horizontal distance between facilities. Manhattan distance is computed as shown in Equation (2).

$$d_{\rho(i,j), \rho(k,l)} = |l - j| + |k - i| \quad (2)$$

For this problem, we have two objective functions: *material handling cost*,  $MHC$  and *closeness ratio*,  $CR$ , and both are calculated with Equation (1). The difference between them is the weight matrix they use, which is different. Notice that these objectives are opposed due to these weight matrices values. Table 1 shows the weight matrix for  $MHC$  on the left and the weight matrix for  $CR$  on the right. Here we see that the weight between A and E facilities for  $MHC$  is

6, and it is -1 for *CR*. On the contrary, the weights between C and F are 1 and -1. So, depending on the values of these weight matrix, the objectives could be opposed.

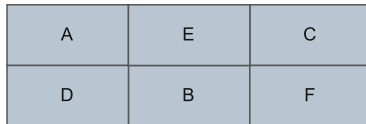
**Table 1.** Weight matrices for the solution in Fig. 1. On the left, the weight matrix for the *MHC*. On the right, the weight matrix for the *CR*.

	A	B	C	D	E	F		A	B	C	D	E	F
A	0	5	3	2	6	4	A	0	4	6	2	-1	4
B	5	0	5	2	6	2	B	4	0	4	2	2	8
C	3	5	0	1	2	1	C	6	4	0	2	2	-1
D	2	2	1	0	2	2	D	2	2	2	0	6	2
E	6	6	2	2	0	6	E	-1	2	2	6	0	1
F	4	2	1	2	6	0	F	4	8	-1	2	1	0

For the sake of understanding we provide an example through Table 1 and Fig. 1. Let us begin with the contribution of the facility located in the first row and in the first column,  $\rho(1, 1) = A$ , to the objective functions. First, we need to calculate the distance between facilities A and D, where  $\rho(1, 2) = D$ . Notice that both facilities are in the same column, so, there is no vertical distance between them ( $d_{AD} = |(1 - 1)| + |(2 - 1)| = 1$ ). Then we proceed with the weight matrix. The contribution to the objective *CR* is  $w_{AD} \cdot d_{AD}$ , where  $d_{AD} = 1$  and  $w_{AD} = 2$ . As a result, we have  $w_{AD} \cdot d_{AD} = 2$ . For the other objective function, we use the weight between A and D. Since  $w_{AD} = 2$ , we have the same result for the objective function *MHC*. We repeat this process for each pairwise between A and the other facilities. In the final step, we calculate the last pairwise facilities value, A and F. For these facilities, where  $\rho(1, 1) = A$  and  $\rho(2, 3) = F$ , the distance is  $d_{AF} = |(3 - 1)| + |(2 - 1)| = 3$ . The weight between facilities A and F is 4, for *CR* and for *MHC*. Then, the contribution of this pairwise for the objective function *CR* is  $w_{AF} \cdot d_{AF} = 12$  and  $w_{AF} \cdot d_{AF} = 12$  for the *MHC* one. In summary, to obtain the total contribution of facility A, we need the following operation:

$$w_{AD} \cdot d_{AD} + w_{AE} \cdot d_{AE} + w_{AB} \cdot d_{AB} + w_{AC} \cdot d_{AC} + w_{AF} \cdot d_{AF}$$

Once we finish calculating facility A, we have to repeat this process with the rest of facilities.



**Fig. 1.** Example layout with size 6.

### 3 Basic Variable Neighborhood Search

One of the most famous metaheuristics is Variable Neighborhood Search (VNS), which was proposed in 1997 by Mladenovic and Hansen [14]. This algorithm has several variants and, among them, one of the most used is the Basic Variable Neighborhood Search (BVNS). This variant lets us escape from the local optima through a *shake* movement. For mono-objective problems, we can apply the schema for BVNS as it was originally proposed. For multi-objective problems, we have to change this schema because this schema works for one solution, but not a set of solutions.

#### 3.1 Bi-Objective BVNS

In this section, we will explain our algorithm proposal and the details of each proposed component for this algorithm.

If we work with mono-objective problems, it is simple to define if one solution is better than another. If we minimize, the solution with lowest value will be the best. If we are maximizing, then the other way around. For multi-objective problems, we have to compare the values for different objective functions. In this particular case, we have two functions that we have to minimize. One solution can dominate another, be dominated by another, or these solutions can be non-dominated among them. More precisely a solution  $\varphi_1$  dominates  $\varphi_2$  ( $\varphi_1 \prec \varphi_2$ ) if the value of objective function  $\mathcal{G}_i(\varphi_1)$  is better or equal for all objectives  $i$ , and exists one objective value where  $\varphi_1$  is better. Equation (3) formally shows this concept.

$$\begin{aligned} & \varphi_1 \prec \varphi_2 \text{ if} \\ & \forall i \in \{1..k\} : \mathcal{G}_i(\varphi_1) \leq \mathcal{G}_i(\varphi_2) \\ & \wedge \exists i \in \{1..k\} : \mathcal{G}_i(\varphi_1) < \mathcal{G}_i(\varphi_2) \end{aligned} \quad (3)$$

Due to the fact that we are going to work with more than one solution, we have to store them in a data structure. We will use a set of non-dominate solutions which we call *ND*. This set will contain only non-dominated solutions. Whenever we want to add a solution  $\varphi$  to this set, we will use the function *Update*, which will check if  $\varphi$  is dominated by any solution in the set. If so, we will not add this solution. If not, we will check all the solutions in the set, removing all the solutions dominated by  $\varphi$ .

One of the most important points of this algorithm is to define when we have improved *our solution*. To this aim, we will use the approach proposed [5]. This approach uses *ND* as the solution to be improved. If any changes have been made to *ND*, we consider that our algorithm has improved our current solution. Algorithm 1 shows our implementation of this approach.

In step 1) we create an empty *ND*. Then we generate a set of solutions by mean of a constructive method in step 2 which will be explained in Sect. 3.2. The number of solutions that we generate is given by the *maxCons* parameter. Then, in step 3, we populate *ND* with the solutions previously generated (*S*). It is worth noting that we *Update* each time we add a solution to our *ND*. Then, in

**Algorithm 1: BVNS ALGORITHM (MAXCONS)**


---

```

1  $ND \leftarrow \emptyset$ 
2  $S \leftarrow \text{Constructive}(\text{maxCons})$  ▷ Section 3.2
3  $ND \leftarrow \text{Update}(ND, S)$ 
4  $S \leftarrow \text{LocalSearch}(ND)$  ▷ Section 3.3
5  $ND \leftarrow \text{Update}(ND, S)$ 
6  $k \leftarrow 1$ 
7 while  $k < kMax$  do
8    $\varphi \leftarrow \text{SelectRandom}(ND)$ 
9    $\varphi' \leftarrow \text{Shake}(\varphi, k)$  ▷ Section 3.4
10   $S \leftarrow \text{LocalSearch}(\{\varphi'\})$  ▷ Section 3.3
11   $ND' \leftarrow \text{Update}(ND, S)$ 
12  if  $(ND \neq ND')$  then
13     $k \leftarrow 1$ 
14     $ND \leftarrow ND'$ ;
15  else
16     $k \leftarrow k + 1$ 
17 return  $ND$ 

```

---

step 4, we try to improve all our solutions as explained in Sect. 3.3. As a result, we obtain a new set  $S$  of improved solutions. Then, we update again our  $ND$  with  $S$ , obtaining the initial set of non-dominated solutions. In step 6, we set  $k = 1$ . Then, in step 7, we will iterate until we reach  $kMax$ . In this case,  $kMax$  depends on the size of the instance. Inside the loop, in step 8, we select a random solution ( $\varphi$ ) from  $ND$ . Then, in step 9, we shake our solution  $\varphi$  as described in Sect. 3.4. The parameter  $k$  indicates how many times we shake our solution. In step 10, we apply our local search procedure to  $\varphi'$ , saving the resulting solutions in  $S$ . In step 11 we update our  $ND$  with  $S$  obtaining a new set  $ND'$ . In step 12 we check if  $ND'$  is equal to  $ND$ . If they are different then there was an improvement. Therefore, we update  $ND$  to  $ND'$  in step 11 and then, in step 13, we set  $k = 1$  to reset the loop. If there was no improvement,  $k$  is incremented in step. Finally, in step 17, we return  $ND$ .

### 3.2 Constructive Method

In this first approach, we have used only one constructive method. With the aim of increasing diversity, our algorithm generates feasible solutions at random following a multi-start approach. This way, the non-dominated solution set will be populated with a number of different solutions. We have run this constructive 100 times, and we have stored these solutions in  $S$ . Once we have these solutions, we proceed to update our  $ND$ . Due to the fact that this set will contain only non-dominated solutions,  $ND$  could contain less than 100 solutions.

### 3.3 Local Search

Once we have a constructive method, we proceed to try to improve the solutions that we have obtained. We have implemented two different neighborhoods through two different moves: interchange and insert. In this section, we will explain the interchange movement. In the next section we will explain the insert one. Moreover, we will explain the local search that we have used. It is worth remembering that the layout is already defined by the instance, that means we know, as a constraint of the problem, the number of rows and the number of facilities per row.

Let us explain the interchange movement through Fig. 2. As stated in Sect. 1, the layout is already defined. Whenever we use a movement, we have to keep the same layout. On the left, we have the original layout, and on the right, we have the resulting one after the movement. The interchange movement changes the position of two facilities, leaving the others in their previous position. As we can see in Fig. 2, we have interchanged facilities D and C.

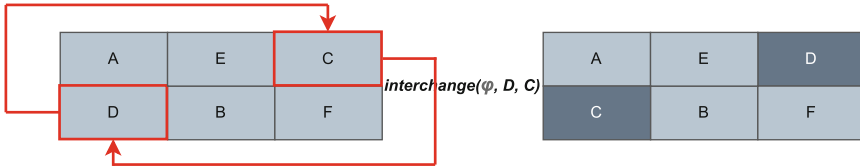


Fig. 2. Interchange facilities C and D.

For the local search, we will use the interchange movement. Our local search is based on the one proposed in [18] which considers only one objective. Due to the fact that we have two objectives, we need to adapt the implementation. The idea is to run two searches on each solution, one per objective. Let us explain in detail the local search through its implementation in Algorithm 2.

First, in step 1 we create an empty initial set of solutions  $S$ . Then in step 2, we will go through all the solutions in  $ND$ . We launch the local search that we mentioned above, focusing on the objective  $MHC$  and store this result in  $s_1$ . We do the same for the  $CR$  in step 4. Then, we add  $s_1$  and  $s_2$  to  $S$  in step 5 and iterate to the next solution in  $ND$ . Finally, we return  $S$  in step 6.

We represent this idea of the alternate local searches in Fig. 3. In this figure, we have two solutions, indicated as (b) and (a). In addition, we have also represented the two objectives  $MHC$  and  $CR$ . On the abscissa axis,  $MHC$  is represented; meanwhile, on the ordinate axis,  $CR$  is represented. There are four colored arrows. On the one hand, the orange ones represent how the solutions follow a trajectory toward the  $MHC$  objective. On the other hand, the blue ones follow a path toward the  $CR$  objective.

---

**Algorithm 2:** LOCALSEARCH(ND)

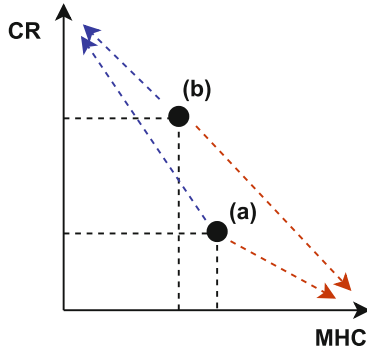
---

```

1  $S \leftarrow \emptyset$ 
2 for  $i = 1$  to  $|ND|$  do
3    $s_1 \leftarrow LS_{MHC}(ND[i])$ 
4    $s_2 \leftarrow LS_{CR}(ND[i])$ 
5    $S \leftarrow S \cup \{s_1\} \cup \{s_2\}$ 
6 return  $S$ 

```

---



**Fig. 3.** Mono-objective local search.

**3.4 Shake**

In the previous section we have explained how the interchange movement works, and how we had used it in our local search procedure. In this section, we explain the insert movement. We have reserved this movement for the shake procedure. The reason is that this movement diversifies more than the interchange movement. Let us explain this movement through Fig. 4.

In Fig. 4 we have the original solution on the left, while on the right we have the one obtained after a certain insert. In particular, we have inserted facility A in the second row, third column. As we can see in this solution, all the other facilities but F have changed positions. The reason is that the layout has been previously defined and we have to keep this layout.



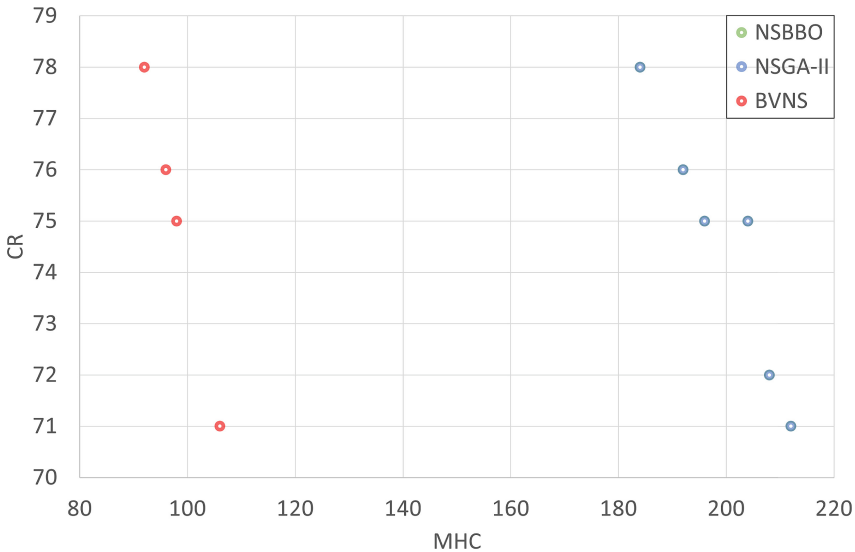
**Fig. 4.** Insert facility A in the second row and third column.

## 4 Results

In this section, we show our experimental results and then compare them with the state of the art. We represent our results graphically, in order to make the comparison easier for the reader.

The previous state of the art presented three algorithms: BBO, NSBBO and NSGA - II [17]. They used BBO as a weighted approach and NSBBO and NSGA - II as a Pareto front approach. In particular, NSBBO is the adaptation of the BBO to the Pareto front approach. The previous authors proposed four instances where we have run our proposal. As it will be shown, the authors reported only a few solutions per instance, but they did not report execution times. The solutions proposed by the previous paper are colored green for NSBBO and blue for NSGA - II. Our proposal is colored red in the figures.

Figure 5 shows the results for the instance of size 6. In this instance, the values reported for NSBBO and NSGA - II are the same. Our BVNS approach obtains four solutions that clearly dominate the previous ones. In fact, solution (71, 105) dominates all previous solutions.



**Fig. 5.** Previous instance with size 6.

Figure 6 shows the results for the instance of size 8. Again, our BVNS approach obtains four solutions that clearly dominate the previous ones.

Figure 7 shows the results for the instance of size 12. In this instance, our BVNS proposal obtains eleven solutions that clearly dominate the previous ones.

Finally, Fig. 8 shows the results for the instance of size 15. In this instance, the values reported for the NSBBO and NSGA - II are more dispersed than in



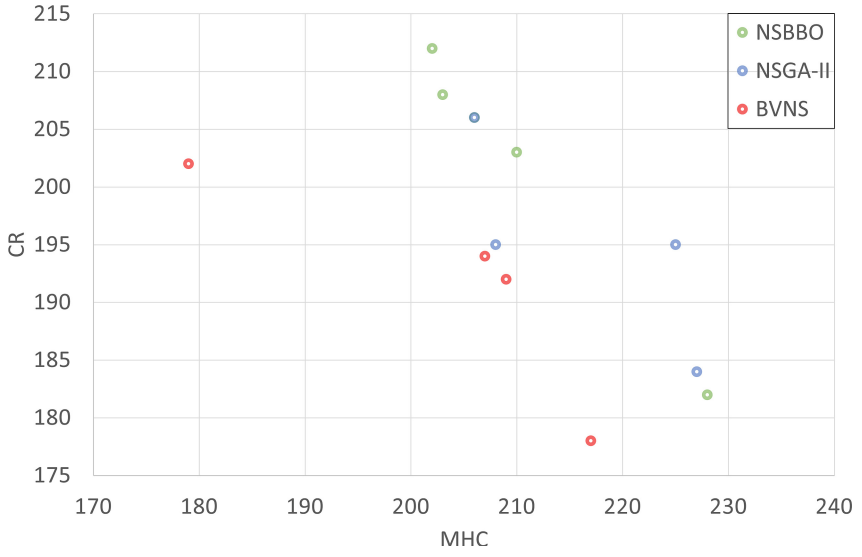


Fig. 6. Previous instance with size 8.

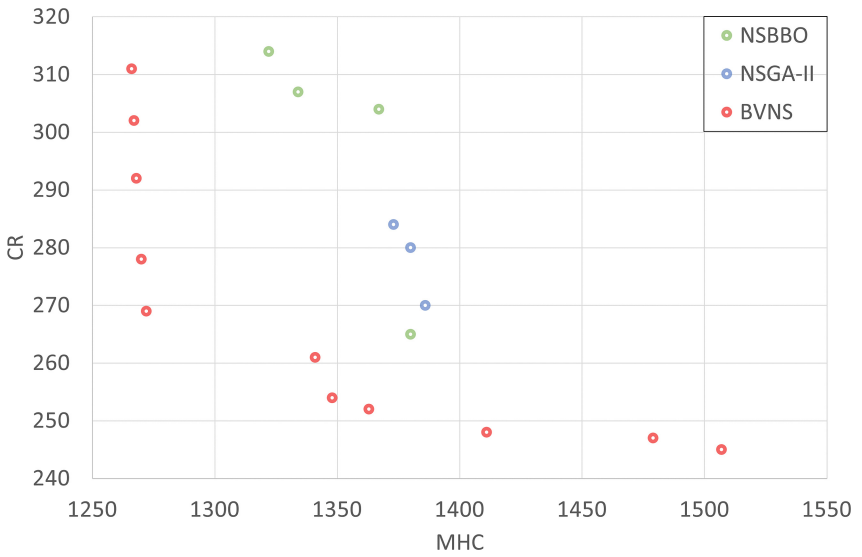
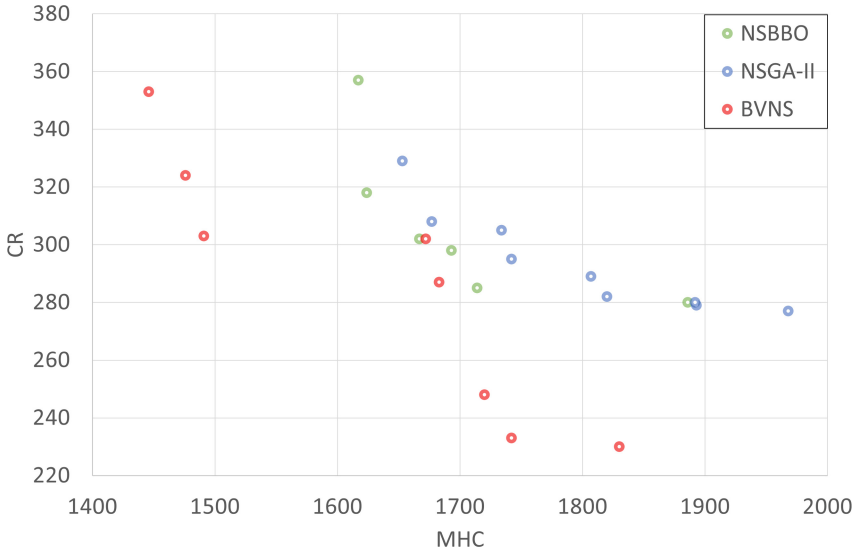


Fig. 7. Previous instance with size 12.

the previous instances. Our BVNS proposal obtains eight solutions where two of them are similar to the previous results. However, the resulting solutions clearly dominates the previous ones.



**Fig. 8.** Previous instance with size 15.

## 5 Conclusions and Future Work

The BO-MREFLP was recently studied in several works, using different approaches. In this paper, we propose a metaheuristic algorithm based on the BVNS methodology. We have reached and improved the results in the state of the art.

As a future work, we will work on a new constructive method based on a GRASP methodology. In addition, it could be possible to implement another kind of local search instead of the mono-objective one. Finally, instead of selecting random solutions from  $ND$ , we can use the crowding distance to select the solution.

## References

1. Ahmadi, A., Pishvaei, M.S., Jokar, M.R.A.: A survey on multi-floor facility layout problems. *Comput. Ind. Eng.* **107**, 158–170 (2017)
2. Anjos, M.F., Fischer, A., Hungerländer, P.: Improved exact approaches for row layout problems with departments of equal length. *Eur. J. Oper. Res.* **270**(2), 514–529 (2018)

3. Chen, C.W.: A design approach to the multi-objective facility layout problem. *Int. J. Prod. Res.* **37**, 1175–1196 (1999)
4. Chung, J., Tanchoco, J.: The double row layout problem. *Int. J. Prod. Res.* **48**(3), 709–727 (2010)
5. Duarte, A., Pantrigo, J., Pardo, E.G., Mladenovic, N.: Multi-objective variable neighborhood search: an application to combinatorial optimization problems. *J. Global Optim.* **63**, 515–536 (2014)
6. Dutta, K.N., Sahu, S.: A multigoal heuristic for facilities design problems: Mughal. *Int. J. Prod. Res.* **20**, 147–154 (1982)
7. Fortenberry, J.C., Cox, J.F.: Multiple criteria approach to the facilities layout problem. *Int. J. Prod. Res.* **23**, 773–782 (1985)
8. Harmonosky, C.M., Tothoro, G.K.: A multi-factor plant layout methodology. *Int. J. Prod. Res.* **30**, 1773–1789 (1992)
9. Hosseini-Nasab, H., Fereidouni, S., Fatemi Ghomi, S.M.T., Fakhrazad, M.B.: Classification of facility layout problems: a review study. *Int. J. Adv. Manuf. Technol.* **94**(1), 957–977 (2018)
10. Hosseini-Nasab, H., Fereidouni, S., Mohammad Taghi Fatemi Ghomi, S., Fakhrazad, M.: Classification of facility layout problems: a review study. *Int. J. Adv. Manuf. Technol.* **94**, 957–977 (2017)
11. Hungerländer, P.: Single-row equidistant facility layout as a special case of single-row facility layout. *Int. J. Prod. Res.* **52**(5), 1257–1268 (2014)
12. Hungerländer, P., Anjos, M.F.: A semidefinite optimization-based approach for global optimization of multi-row facility layout. *Eur. J. Oper. Res.* **245**(1), 46–61 (2015)
13. Matai, R., Singh, S., Mittal, M.: Modified simulated annealing based approach for multi objective facility layout problem. *Int. J. Prod. Res.* **51**, 4273–4288 (2013)
14. Mladenović, N., Hansen, P.: Variable neighborhood search. *Comput. Oper. Res.* **24**(11), 1097–1100 (1997)
15. Rosenblatt, M.J.: The facilities layout problem: a multi-goal approach. *Int. J. Prod. Res.* **17**, 323–332 (1979)
16. Simmons, D.M.: One-dimensional space allocation: an ordering algorithm. *Oper. Res.* **17**(5), 812–826 (1969)
17. Singh, D., Ingole, S.: Multi-objective facility layout problems using BBO, NSBBO and NSGA-II metaheuristic algorithms. *Int. J. Ind. Eng. Comput.* **10**(2), 239–262 (2019)
18. Uribe, N.R., Herrán, A., Colmenar, J.M., Duarte, A.: An improved GRASP method for the multiple row equal facility layout problem. *Expert Syst. Appl.* **182**, 115184 (2021)