A Path Relinking-based approach for the Bi-Objective Double Floor Corridor Allocation Problem

Nicolás R. Uribe[1][0000−0001−8984−7715], Alberto Herrán[2][0000−0003−0348−0313], and J. Manuel Colmenar[3][0000−0001−7490−9450]

Universidad Rey Juan Carlos
Department of Computer Science and Statistics, Spain
{nicolas.rodriguez, alberto.herran, josemanuel.colmenar}@urjc.es

Abstract. The Bi-Objective Double Floor Corridor Allocation Problem is one of the most recent incorporation to the family of Facility Layout Problems. This problem, which has been a challenge for exact and metaheuristic approaches, involves optimizing the layout of the given facilities to minimize material handling cost and the length of the corridor considering more than one floor. This paper introduces a new approach based on the combination of two greedy methods and a path relinking implementation to tackle this problem. The experimental results show the superiority of our proposal in relation to the current state-of-the-art under different multi-objective metrics.

Keywords: Path Relinking · Bi-Objective optimization · Facility Layout Problem

1 Introduction

Facility Layout Problems (FLP) aim to optimize facility arrangements in order to minimize a certain objective function. This family of problems has a wide range of applications, such as manufacturing, delivery services, urban planning, and computer storage design [11]. Typically, three resolution approaches are found in the literature [2]: exact, heuristics, and machine learning approaches. Initial FLP research began with the Single Row Facility Layout Problem (SRFLP) [13], followed by the Double Row (DRFLP) [3] and Multiple Row (MRFLP) [7] versions, each with different row layouts and conditions. A significant variant is the Corridor Allocation Problem (CAP) [1], which is a DRFLP variant without spaces between facilities.

Previous papers dealt with one objective, the material handling cost (MHC). Recently, new variants of the problem have been studied where an additional objective is considered, such as the closeness rating (CR) [14] or the corridor length (CL) [8]. The Bi-Objective Corridor Allocation Problem (bCAP) [9] and Bi-Objective Double Floor Corridor Allocation Problem (bDFCAP) [6] involve optimizing facility layout for MHC and CL. The latter is a two-floor variant

of bCAP, which was tackled by means of a Mixed Integer Linear Programming and a Memetic Algorithm (combining a Genetic Algorithm and Variable Neighborhood Search) in the previous work, providing results for instances up to size 30. In this paper, we study the bDFCAP proposing a metaheuristic algorithm based on the combination of a greedy constructive method and Path Relinking. This approach has obtained competitive results in relation to the state-of-the-art method in the studied instances.

The remaining sections of this paper are organized as follows. In Section 2, we present the description of the problem. In Section 3, our optimization proposal is described. In Section 4, we provide an analysis of our results and compare them with the state of the art. Finally, in Section 5, we present our conclusions and future work.

2 Problem description

The bDFCAP considers a layout with two floors: the lower and the upper ones. In each floor, facilities can be located at both sides of a corridor without allowing any gap between two adjacent facilities in a row. In addition, there is an elevator on the side where the origin of the four rows is set, allowing the flow of material between facilities located on different floors (see Fig. 1, where the origin is set on the left-hand side). The objective is to arrange all the facilities in the layout minimizing both, the overall *MHC*, defined as the weighted sum of the center-to-center distances between each pair of facilities in the layout, and the *CL*, defined as the length of the longest row. Notice that the distance between facilities in different floors must consider the route through the elevator.

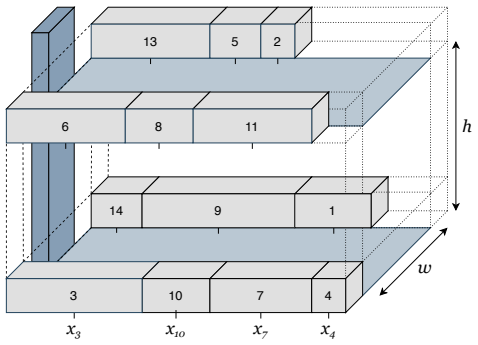

Fig. 1. Layout with two floors and one corridor per floor. Facility 3 is located in the first row of the first floor ($f_3 = 1$, $r_3 = 1$). Similarly, facility 14 has ($f_{14} = 1$, $r_{14} = 2$), facility 6 has ($f_6 = 2$, $r_6 = 1$), facility 13 has ($f_{13} = 2$, $r_{13} = 2$), and so on.

More formally, given a set F of n facilities, $n = |F|$, where each $i \in F$ has an associated length l_i; the flow cost per unit distance c_{ij} between each pair of

facilities $i, j \in F$; and a layout with two floors (floors 1 and 2, separated by a height h), and two rows in each floor (rows 1 and 2, separated by a corridor of width w); the bDFCAP consist in finding an assignment of facilities to floors $f : F \to \{1, 2\}$ and rows $r : F \to \{1, 2\}$, and a vector $x \in \mathbb{R}^n$ with the center positions of all the facilities in the layout (measured from the common fixed left origin where the elevator is located) that minimizes both the total *MHC* and *CL*. Mathematically:

$$\min \ \mathcal{F}(f, r, x) = \{\mathcal{F}_{MHC}, \mathcal{F}_{CL}\} \tag{1a}$$

$$\text{s.t. } \mathcal{F}_{MHC} = \sum_{\substack{i,j \in F \\ i<j}} c_{ij} d_{ij} \tag{1b}$$

$$\mathcal{F}_{CL} = max(L_{11}, L_{12}, L_{21}, L_{22}) \tag{1c}$$

$$|x_i - x_j| \geq (l_i + l_j)/2 \qquad i, j \in F, \ i < j, \ r_i = r_j \tag{1d}$$

$$d_{ij} = |x_i - x_j| \qquad i, j \in F, \ i < j, \ r_i = r_j \tag{1e}$$

$$d_{ij} = |x_i - x_j| + w \qquad i, j \in F, \ i < j, \ f_i = f_j, \ r_i \neq r_j \tag{1f}$$

$$d_{ij} = x_i + x_j + w + h \qquad i, j \in F, \ i < j, \ f_i \neq f_j \tag{1g}$$

$$L_{ab} = \sum_{\substack{i \in F \\ f_i = a \wedge r_i = b}} l_i \qquad a, b \in \{1, 2\} \tag{1h}$$

Equations (1b) and (1c) represent the *MHC* and *CL* objectives, respectively. Equation (1d) avoids the overlapping between two adjacent facilities in the same row. Equations (1e) to (1g) compute the distance between two facilities in three different situations: (1e) located in the same row; (1f) located in different rows of the same floor; and (1g) located in different floors. Finally, Equation (1h) computes the length of each row.

3 Optimization proposal

This paper introduces a Path Relinking (PR) approach to tackle the bDFCAP problem. Originally conceptualized as a method to combine intensification and diversification strategies within Tabu Search [12], PR is based on the notion of establishing a trajectory between two solutions. The aim is to discover potential solutions while traversing this trajectory. The process involves incremental integration of the features of a second high-quality solution, known as the guide solution, into a first solution, known as the initial solution. Given that both solutions are of considerable quality, the expected outcome is that exploration along the generated path will venture into new and valuable areas of the search space.

3.1 Bi-Objective PR

In single-objective problems, determining the superiority of one solution over another is straightforward. For minimization problems, the solution with the lowest

value is preferable, while in maximization problems, the highest value prevails. However, in multi-objective problems, the comparison involves multiple objective functions. Specifically, in this context, we are dealing with two functions that need to be minimized. A solution can either dominate, be dominated by, or be non-dominated with respect to another. To be more precise, a solution φ_1 is said to dominate another solution φ_2 (denoted as $\varphi_1 \prec \varphi_2$) if for every objective function $\mathcal{F}_i$, φ_1 is either better or equal, and there is at least one objective function where φ_1 is better. This concept is formally defined in Equation (2).

$$\begin{aligned} \varphi_1 \prec \varphi_2 \; if \\ \forall i \in \{1..k\} : \mathcal{F}_i(\varphi_1) \leq \mathcal{F}_i(\varphi_2) \\ \wedge \exists i \in \{1..k\} : \mathcal{F}_i(\varphi_1) < \mathcal{F}_i(\varphi_2) \end{aligned} \tag{2}$$

Since our algorithm deals with multiple solutions at the same time, they must be organized within a suitable data structure. For this purpose, we will utilize a set named as *ND*, designated for storing only non-dominated solutions. To incorporate a new solution φ into this collection, we will employ an `Update` function. This function will first determine whether φ is dominated by any existing member of the set. Should φ not be dominated, the function will proceed to evaluate all present solutions within the set, excluding any dominated by φ.

3.2 Path Relinking

Our PR proposal creates a path between two solutions φ and χ by iteratively including in φ elements from χ. The method starts using *insert* moves to balance the rows from the initial solution to the guide solution. Then, it applies *interchange* moves to match the guide solution. Moreover, the method tries to update the *ND* set with all the solutions generated in the path.

Fig. 2 shows an example of the whole procedure using the instance `S9H`, where an initial solution φ (in the top part of the figure) will be modified until reaching the guide solution χ (at bottom). Our idea behind the *insert* move is to ensure that each row in the initial solution matches the size of its corresponding row in the guide solution. For this purpose, let us define a vector N_φ with the number of facilities in each row of φ, hence, in this example, $N_\varphi = \{4, 3, 1, 1\}$ and $N_\chi = \{2, 2, 3, 2\}$. Then, in order to balance the number of facilities in each row of φ, all the candidate *insert* moves at each iteration are those insertions of facilities from a row $row1$ with $N_\varphi(row1) > N_\chi(row1)$ to a row $row2$ with $N_\varphi(row2) < N_\chi(row2)$. In this example, facilities 6, 2 and 4 in $row1$ of φ (not present in $row1$ of χ) are candidate facilities to be inserted in rows 3 and/or 4. Similarly, the three facilities in $row2$ of φ (not present in $row2$ of χ) can be removed but, in this case, only facilities 1 and 8 can be inserted in $row3$. Fig. 2 shows below the initial solution two *insert* moves among the six possible moves. Moreover, we intend to benefit from placing a facility in the identical location as it appears in the guide solution (facility 6 inserted at the beginning of $row3$, and facilities 2 and 4 at the end or beginning of $row4$, respectively). Once we

have all the solutions resulting after the possible insertions at each iteration (6 in this case), the method selects one of the solutions at random to belong to the path and continue the procedure. The selected solutions, which belong to the path, are highlighted in red in the figure. This phase of the process continues until each row in the initial solution matches the size of its corresponding row in the guide solution, ending in what we call the intermediate solution.

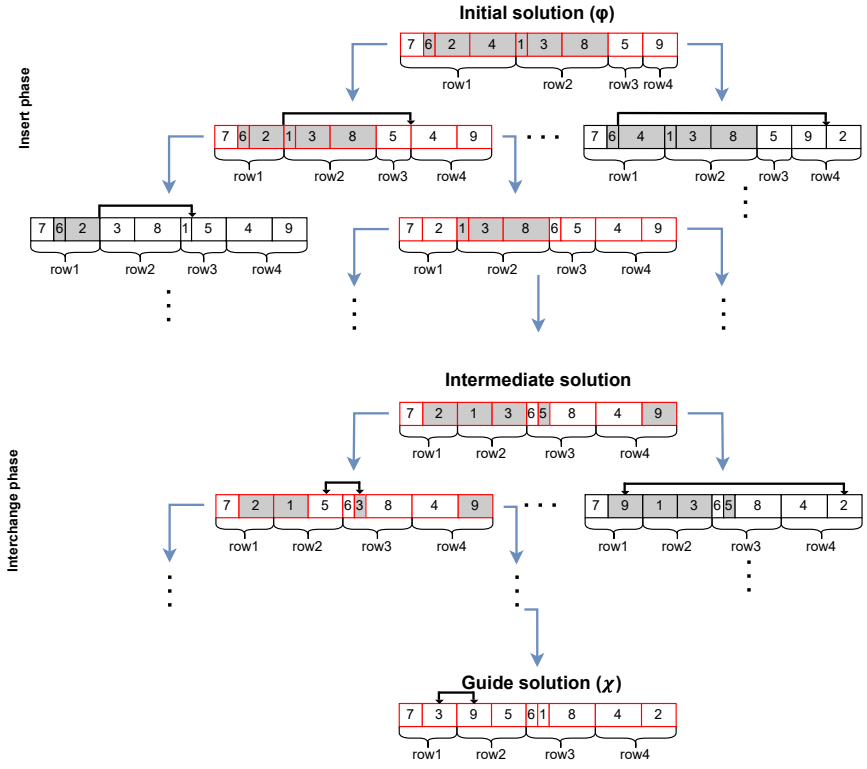

Fig. 2. Graphic representation for the `PathRelinking` procedure.

The second phase of the PR procedure compares the intermediate solution with the guide solution to match the position of all the facilities in both solutions. Therefore, it analyzes all the facilities in φ to check if they are located at the same position in χ, otherwise an interchange is needed to match this facility in both solutions. Following the example in Fig. 2, only five facilities from the intermediate solution (2, 1, 3, 5 and 9) can exchange its position in φ at the first iteration of the second phase. Then, the procedure selects one of these moves using a given function, following a Greedy Randomized Adaptive Search Procedure (GRASP) methodology [5]. In this case, the greedy function of a candidate move $g(move)$ is the objective function of the resulting solution after

the interchange move, and the selected move is randomly chosen from a restricted candidate list built including all the candidate moves with $g(move) \leq g_{min} + \alpha \cdot (g_{max} - g_{min})$. Hence, the randomness/greediness of the procedure is controlled by the α parameter ($\alpha = 0$ purely random, $\alpha = 1$ purely greedy). Once the move is applied, the resulting solution is added to the path, and the procedure continues after it.

We have encapsulated the whole PR procedure in a method `PathRelinking`(φ, χ, α, *func*), where φ is the initial solution, χ is the guide solution, α is the parameter required by the GRASP methodology of the interchange phase, and *func* is the objective function (*MHC* or *CL*) to be used as the greedy function by the GRASP. The method will return the set of solutions selected for the path.

3.3 Algorithmic description

Algorithm 1 shows the pseudo-code of our Bi-objective Path Relinking (BPR) proposal. The algorithm receives two input parameters: the number of iterations for the greedy construction phase, *maxCons*; and a value α controlling random/greedy selection of the move to be performed at each iteration of the `PathRelinking` procedure.

In step 1, we initialize our set of non-dominated solutions, called *ND*, to an empty set. In step 2, we generate *maxCons* solutions with the `Greedy` method considering *MHC* and store them in S_1. In step 3, we repeat the same process but considering *CL*. This greedy algorithm will be later explained. Next, in steps 4 to 7, we proceed with a `PathRelinking` process between each solution φ from S_1 and each solution χ from S_2 (steps 6 and 7) and then the other way around (steps 8 and 9). Notice that the first and third methods consider *MHC*, while the second and fourth consider *CL*. These steps generate the initial set of non-dominated solutions *ND*. In step 10, we update *ND* with the sets of non-dominated solutions generated in steps 6 to 9. Notice that the value for the third parameter in the uses of `PathRelinking` is 1 in these steps. The reason of this value is that we want the most greedy behavior in this phase. In step 11, we set *improve* to true, and then, in step 12, we enter a loop. The aim of this loop is to improve *ND* while new solutions are included in *ND*. In step 13, we set *improve* to false. In step 14 to step 19 we repeat a similar procedure for the `PathRelinking`, but instead of using S_1 and S_2, using the solutions in *ND*. Notice that in step 20 we Update *ND'* instead of the original *ND*. In step 21 if *ND'* and *ND* are different, we set *ND* to *ND'* in step 23. Finally, in step 24, we return the final set of non-dominated solutions *ND*.

Regarding the `Greedy` algorithm, it generates *maxCons* solutions based on a greedy strategy. Firstly, it selects four random facilities and places them in each row. Then, the remaining facilities are placed in the solution considering the indicated objective function as greedy criteria, either *MHC* or *CL*.

Algorithm 1: BPR($maxCons, \alpha$)

```
1  ND ← ∅
2  S_1 ← Greedy(maxCons, MHC)
3  S_2 ← Greedy(maxCons, CL)
4  for φ ∈ S_1 do
5      for χ ∈ S_2 do
6          P_1 ← PathRelinking(φ, χ, 1, MHC)
7          P_2 ← PathRelinking(φ, χ, 1, CL)
8          P_3 ← PathRelinking(χ, φ, 1, MHC)
9          P_4 ← PathRelinking(χ, φ, 1, CL)
10         ND ← Update(ND ∪ P_1 ∪ P_2 ∪ P_3 ∪ P_4)
11 improve ← true
12 while improve do
13     improve ← false
14     for i = 1 to |ND| − 1 do
15         for j = i + 1 to |ND| do
16             P_1 ← PathRelinking(ND[i], ND[j], α, MHC)
17             P_2 ← PathRelinking(ND[i], ND[j], α, CL)
18             P_3 ← PathRelinking(ND[j], ND[i], α, MHC)
19             P_4 ← PathRelinking(ND[j], ND[i], α, CL)
20             ND′ ← Update(ND ∪ P_1 ∪ P_2 ∪ P_3 ∪ P_4)
21     if (ND ≠ ND′) then
22         improve ← true
23         ND ← ND′
24 return ND
```

4 Results

In this section, we present our experimental findings and subsequently benchmark them against the state of the art. We provide our results through various metrics to facilitate a clearer comparison. The parameter values used in the execution of our algorithm are the following: $maxCons = 5 \cdot n$, where n is the size of the instance, and $\alpha = RND$, where RND means that there is a random value [0, 1] for each iteration.

This research evaluates the results using literature multi-objective metrics [10] [15], where each metric calculates different values, such as the dominance, the distance between solutions, etc. Except for the Hypervolume, these metrics require a comparison between two sets of non-dominated solutions. For each problem instance, we compiled a *reference set* of non-dominated solutions by aggregating the results from the state-of-the-art and our algorithm.

The coverage metric $C(X, Y)$ assesses the proportion of solutions from algorithm X that weakly dominate those from algorithm Y, with $C(X, Y) = 1$ suggesting that all solutions from Y are weakly dominated by those from X. We define coverage as $C(Ref, Alg)$, where Ref is the reference set. Hypervolume (HV) measures the objective space volume occupied by a set of non-dominated solutions. The epsilon metric (ϵ) quantifies the minimum distance needed to reach solutions from a set of non-dominated solutions to the reference set. Generational (GD), inverse generational (IGD), and additive inverse generational ($IGD+$) distances quantify the divergence of solutions from the reference set, each with its distinct, but similar, calculation methodology. We also considered the number of solutions in each set, labeled as *Size*, and the spread (Δ), which are the size of the set, and the average distance between adjacent solutions, respectively. Also, we compare the computational time in seconds (T(s)).

These metrics were calculated using the *jMetal* [4] framework. For all metrics but HV and *Size*, lower values indicate better results.

We compare our algorithm proposal, named *BPR*, with the algorithm in [6]. The authors propose a memetic algorithm, composed by a Genetic Algorithm with Variable Neighborhood Search (GAVNS). Table 1 displays the performance of both algorithms. There are 22 instances in the state of the art, and we have split them in 4 sets, depending on the size of the instance, for a better comparison. In addition, the total average is calculated for each metric. In the first set, our proposal obtains better results for the $HV, GD, Size$ and time. In the second set, we obtain better results for the $GD, Size$ and Δ. It is the only set in which we obtain HV worse than GAVNS and better results in Δ. For the third and fourth sets, our approach reaches better results in $HV, \epsilon, GD, Size$, and $T(s)$. It is important to note that we have spent only 12%, 6%, 16% and 20% of the execution time of the state of the art, respectively. In conclusion, we obtain better results in HV, ϵ, GD and $Size$, spending only 20% of the execution time of the GAVNS on average.

In addition, we also include Fig. 3 as a graphic representation of the results for both algorithms in instance `N30_05`. In blue, we have the results for the GAVNS algorithm, and in red, for our proposal. In the ordinates, the values

Set	Algorithm	C(Ref, Alg)	HV	ϵ	GD	IGD	IGD+	Size	Δ	T(s)
[9, 12]	BPR	0.24	**0.49**	0.10	**1227.35**	1088.17	3000.07	**7**	1.00	**4.67**
	GAVNS	**0.02**	0.47	**0.09**	1304.81	**1069.81**	**2955.10**	6	**0.98**	38.45
[13, 20]	BPR	0.52	0.49	0.12	**1781.97**	1548.09	5699.38	**12**	**0.98**	**48.33**
	GAVNS	**0.04**	**0.52**	**0.11**	1943.87	**1525.20**	**5616.89**	10	0.99	773.89
[25]	BPR	0.43	**0.41**	**0.04**	**3178.81**	2958.89	14203.25	**23**	0.95	**443.40**
	GAVNS	**0.10**	0.39	0.09	5067.48	**2939.09**	**14118.05**	9	**0.93**	2803.09
[30]	BPR	0.52	**0.37**	**0.10**	**6525.34**	5975.67	26476.98	**18**	0.96	**1227.80**
	GAVNS	**0.13**	0.34	0.12	9019.89	**5912.30**	**26193.00**	9	**0.95**	4967.20
Total	BPR	0.42	**0.45**	**0.09**	**3026.21**	2749.56	11618.08	**14**	0.98	**394.27**
	GAVNS	**0.07**	0.43	0.10	4087.68	**2719.41**	**11499.42**	8	**0.96**	1987.52

Table 1. Overview for both algorithms for each set and each metric.

for *CL* are represented, while in the abscissas, the values for *MHC*. It is worth mentioning that GAVNS obtains better values in both ends, mainly in the end for *MHC*, where our proposal is still far from those solutions. However, our algorithm obtains non-dominated solutions in the middle of the front. Although we have represented one of the largest solutions, the performance is similar in the other instances.

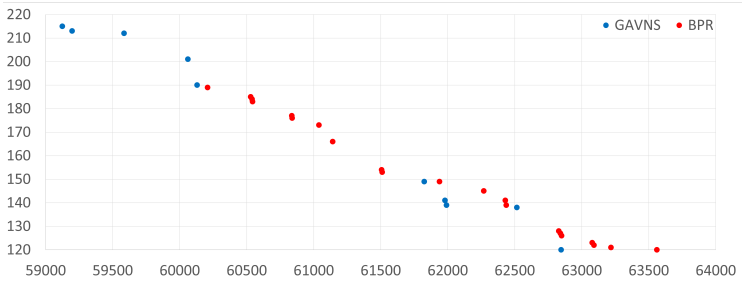

Fig. 3. Representation of the results for both algorithms in instance `N30_05`.

5 Conclusions and future work

The Bi-Objective Double Floor Corridor Allocation Problem has been introduced in the recent literature. This paper details the implementation of a metaheuristic algorithm that utilizes the Path Relinking technique as alternative to the state of the art. Our proposal obtained competitive results in eight different multi-objective metrics spending a 20% of the time of the previous work.

Looking ahead, our research will focus on incorporating an External Path Relinking technique in order to improve the search on the ends of the front, where the previous approach obtains a good behavior. Moreover, our aim is to enhance our findings by improving aspects like coverage or execution time.

Acknowledgements

This work has been partially supported by the Spanish Ministerio de Ciencia e Innovación (MCIN/AEI/10.13039/501100011033) under grant refs. RED2022-134480-T, PID2021-126605NB-I00 and by ERDF A way of making Europe; and Generalitat Valenciana with grant ref. CIAICO/2021/224.

References

1. Amaral, A.R.: The corridor allocation problem. Computers & Operations Research **39**(12), 3325 – 3330 (2012)
2. Burggräf, P., Wagner, J., Heinbach, B.: Bibliometric study on the use of machine learning as resolution technique for facility layout problems. IEEE Access **9**, 22569–22586 (2021). https://doi.org/10.1109/ACCESS.2021.3054563
3. Chung, J., Tanchoco, J.: The double row layout problem. International Journal of Production Research **48**(3), 709–727 (2010). https://doi.org/10.1080/00207540802192126
4. Durillo, J.J., Nebro, A.J.: jmetal: A java framework for multi-objective optimization. Advances in Engineering Software **42**(10), 760–771 (2011)
5. Feo, T., Resende, M.: A probabilistic heuristic for a computationally difficult set covering problem. Operations Research Letters **8**, 67–71 (1989)
6. Guan, C., Zhang, Z., Gong, J., Liu, S.: Mixed integer linear programming model and an effective algorithm for the bi-objective double-floor corridor allocation problem. Computers & Operations Research **132**, 105283 (2021). https://doi.org/10.1016/j.cor.2021.105283
7. Hungerländer, P., Anjos, M.F.: A semidefinite optimization-based approach for global optimization of multi-row facility layout. European Journal of Operational Research **245**(1), 46 – 61 (2015). https://doi.org/10.1016/j.ejor.2015.02.049
8. Kalita, Z., Datta, D.: Solving the bi-objective corridor allocation problem using a permutation-based genetic algorithm. Computers & Operations Research **52**, 123–134 (2014). https://doi.org/10.1016/j.cor.2014.07.008
9. Kalita, Z., Datta, D., Palubeckis, G.: Bi-objective corridor allocation problem using a permutation-based genetic algorithm hybridized with a local search technique. Soft Computing **23**(3), 961–986 (Feb 2019). https://doi.org/10.1007/s00500-017-2807-0
10. Knowles, J.D., Thiele, L., Zitzler, E.: A tutorial on the performance assessment of stochastic multiobjective optimizers, vol. 214. ETH Zurich (2006)
11. Pablo Pérez-Gosende, J.M., Díaz-Madroñero, M.: Facility layout planning. an extended literature review. International Journal of Production Research **59**(12), 3777–3816 (2021). https://doi.org/10.1080/00207543.2021.1897176
12. Resende, M.G., Ribeiro, C.C.: Scatter Search and Path Relinking: Advances and Applications, pp. 1–37. Springer US, Boston, MA (2010)
13. Simmons, D.M.: One-dimensional space allocation: An ordering algorithm. Operations Research **17**(5), 812–826 (1969)
14. Singh, D., Ingole, S.: Multi-objective facility layout problems using BBO, NSBBO and NSGA-II metaheuristic algorithms. International Journal of Industrial Engineering Computations pp. 239–262 (2019)
15. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation **7**(2), 117–132 (2003)