

An Iterated Greedy with Path Relinking for Corridor Allocation Problem

Rafael F. R. Corrêa¹, Luciana B. Gonçalves², Stênio S. R. F. Soares²,
Lorenza L.O. Moreno², Heder S. Bernardino², João M. Freitas³

Universidade Federal de Juiz de Fora (UFJF)

Juiz de Fora, Brazil

rafael.freesz@estudante.ufjf.br¹
{lbrugiollo, stenio.soares, lorenza,
heder.bernardino}@ufjf.br²
jmfreitas.ai@gmail.com³

ABSTRACT

Layouts are widely applied in real-world scenarios, such as arranging rooms in buildings, machinery in production lines, and semiconductors on circuit boards. A well-designed layout is crucial for minimizing costs and increasing efficiency. The Corridor Allocation Problem (CAP) focuses on arranging facilities along a two-sided corridor to minimize communication costs. A previous study applied a Grammar-based Genetic Programming Hyper-heuristic to CAP, where the best-generated heuristic resembled an Iterated Greedy (IG) algorithm, highlighting its potential. Motivated by this, we propose a dedicated IG algorithm for solving CAP, designing each component to fully explore this technique's capabilities. Our method incorporates Path Relinking (PR) and Random Variable Neighborhood Descent as local search strategies, resulting in the IG-PR algorithm. We tested our approach on 89 benchmark instances and obtained competitive results compared to state-of-the-art methods, achieving similar or better solution quality while requiring less execution time than the best-performing algorithm reported in the literature.

KEYWORDS. Corridor Allocation Problem, Iterated Greedy, Path Relinking, Random Variable Neighborhood Descent

AD&GP, AdP&ED, OA

1. Introduction

Reducing costs and increasing efficiency are important goals in industry, engineering, and science. The positioning of rooms in a building under construction, machinery in production lines, and semiconductors on printed circuit boards are examples of contexts that directly affect these factors. These are known as layout problems, which deal with the physical arrangement of facilities along a given area, forming a layout [Amaral, 2012; Cellin, 2017]. A well-designed layout can achieve cost reductions, improve work efficiency, and minimize the required space. Thus, the competitiveness of enterprises can be increased by optimizing their layouts. Several layout problems have been identified in the literature. In this work, we deal with the Corridor Allocation Problem (CAP) [Amaral, 2012], an extension of the Double-Row Facility Layout Problem (DRFLP) [Chung and Tanchoco, 2010].

The Double-Row Facility Layout Problem (DRFLP) deals with the arrangement of facilities along two rows. In this scenario, each pair of facilities (i, j) has a cost that is defined by the product of the demand coefficient $c_{i,j}$ between them and their distance $d_{i,j}$, where the interpretation of the demand coefficient depends on the context of the problem to be solved. Thus, the total cost of a layout is defined as the sum of the costs of all arranged facility pairs. For instance, it may represent the volume of demand, the average daily traffic, and the priority in functional dependencies. The only constraint of this problem is that it does not allow overlapping facilities along the corridor.

In the definition of the Corridor Allocation Problem (CAP) in [Amaral, 2012], two additional constraints are imposed: (i) empty spaces between facilities are not allowed, and (ii) the first facility on each side must be placed at the beginning of the corridor, on the left. Figure 1 shows a proper example of CAP.

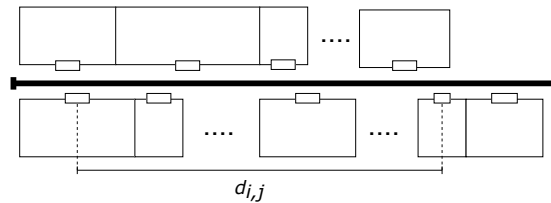


Figure 1: Illustrative example of a CAP. Extracted from [Correa et al., 2022].

Layout problems are NP-hard [Herrán et al., 2021]. In practice, heuristics are applicable for dealing with instances of relevant size, even though there exist exact methods capable of obtaining the optimal solution for small-sized instances.

Several papers related to CAP approach the problem using exact methods and metaheuristics. Initially, in [Amaral, 2012], a Mixed-Integer Programming (MIP) is proposed using 2-opt and 3-opt algorithms as local search. Optimal results were performed for 13 instances of size $9 \leq n \leq 30$, where n is the number of facilities. The execution times shown were high. Two evolutionary algorithms were presented in [Ghosh and Kothari, 2012]. The first refers to a Genetic Algorithm (GA) allied to a local search. A Scatter Search (SS) with Path Relinking (PR) was presented in a second approach. A total of 45 instances of size $9 \leq n \leq 49$ were used, achieving the same results as the previous work with a considerably reduced execution time. Simulated Annealing (SA) and Tabu Search (TS) were used in [Ahonen et al., 2014]. In that work, 68 instances of size $9 \leq n \leq 70$ were used, presenting better results than the previous work, with reduced time. The most recent work in the literature presents a Parallel Basic Variable Neighborhood Search (PBBVNS) [Herrán et al., 2021] and a Clonal Selection Algorithm (CLONALG) [Corrêa et al., 2024], using a total of 89 instances of size $9 \leq n \leq 70$. To the best of our knowledge, the best results currently available are those of PBBVNS and CLONALG. While PBBVNS has achieved the best-known

results in the literature, CLONALG produces good solutions with significantly lower computational time.

The design of heuristics tends to be complex due to a large number of user-defined parameters and the definition of their appropriate components. Hyper-heuristics deal with these difficulties by automating the heuristic development process [Burke et al., 2013]. Therefore, a Grammar-based Genetic Programming Hyper-heuristic (GGPHH) was proposed in [Correa et al., 2022] to automatically develop heuristics for CAP. The best heuristic generated presents an iterative sequence of partial destruction and reconstruction of a given solution, followed by local searches, resembling the behavior of an Iterated Greedy, metaheuristic proposed in [Ruiz and Stützle, 2007]. The generated heuristic was applied to a set of 89 instances and achieved the best result and the best average of results in 30.

As can be seen, the results from [Correa et al., 2022] are promising. However, the main role of hyper-heuristics is not to directly deliver optimized solutions, but to generate high-level strategies by combining low-level heuristics in a generalizable way. In this scenario, the best-performing algorithm suggested by the hyper-heuristic includes redundant local search operations, which compromise its efficiency. Moreover, its capabilities are inherently limited to the heuristics provided by the grammar. Given these limitations, and to explore the potential of the Iterated Greedy (IG) framework, we propose a dedicated IG algorithm specifically designed for solving the CAP. Each component of the algorithm is designed to improve the performance of this technique, resulting in a more effective, elaborate, and robust approach.

Thus, to evolve the suggested heuristic in [Correa et al., 2022], we propose the IG-PR, an algorithm that combines Iterated Greedy (IG) with Path Relinking and Random Variable Neighborhood Descent (RVND) as local search techniques. To the best of our knowledge, there is no work in the literature that applies IG to solve the Corridor Allocation Problem.

The remainder of this paper is organized as follows. Section 2 formally describes the CAP. Section 3 presents the methods used. Section 4 presents the IG developed in this work. Section 5 presents the results of the experiments. Finally, Section 6 presents the conclusions and suggestions for future work.

2. Problem Description

The CAP was proposed by [Amaral, 2012] as a variant of DRFLP. The problem deals with the distribution of a set F of n facilities along a two-sided corridor, as shown in Figure 1. Each facility has a length $l_i, \forall i \in F$. Each pair of facilities has a demand coefficient $c_{i,j}, \forall i, j \in F$. The meaning of $c_{i,j}$ is defined by the practical context to which it is applied. In general, it can be understood as the intensity of dependence between a pair of facilities. Also, the following constraints must be attended to:

1. Facility overlaps are not allowed in the layout;
2. Both sides start at the same point (i.e., left-aligned rows);
3. Empty spaces are not allowed in consecutive pairs of facilities.

The abscissa x_i of a facility $i, \forall i \in F$ corresponds to the distance between the beginning of the corridor and its midpoint. The distance between any pair of facilities is given by the absolute value of the difference of their midpoints, as $d_{i,j} = |x_i - x_j|, \forall i, j \in F$.

The cost of a layout is given by the sum of the costs of the pairs of facilities. Also, the cost of a pair of facilities i and j is given by the product of the distance $d_{i,j}$ by their demand coefficient $c_{i,j}$. Thus, the objective of CAP is to minimize this sum of costs, given by

$$\min \sum_{i \in F} \sum_{j \in F, i < j} c_{i,j} \times d_{i,j} \quad (1)$$

3. Methods

The present paper proposes an approach that employs different techniques to deal with the CAP. These techniques are presented in this section.

3.1. Randomized Greedy Algorithm

Randomized Greedy Algorithm (RGA) is widely adopted in optimization problems and incorporates random processes into greedy algorithms [Gao et al., 2018]. While the greedy algorithm constructs a solution by selecting at each iteration the best choice accessible at the time, the RGA selects an element randomly from a list of the best candidates. For a conveniently ordered candidate list L , RGA randomly selects one of the first $\alpha \times |L|$ elements, where $\alpha \in [0, 1]$. When $\alpha = 0.0$, the algorithm is said to be purely greedy. For $\alpha = 1.0$, the algorithm is purely random.

3.2. Random Variable Neighborhood Descent

The RVND [Santos and Arroyo, 2017] is a local search technique. It is a heuristic derived from Variable Neighborhood Descent (VND) [Duarte et al., 2018]. Different local search strategies allow the exploration of different regions of the search space. The purpose of VND is to explore an arbitrary number of local search neighborhoods in a predefined sequence, allowing the exploration of broader regions of the search space. For RVND, the sequence in which the local search neighborhoods are performed is randomly defined in each run [Santos and Arroyo, 2017].

3.3. Path Relinking

The PR technique, proposed in [Glover, 1997], explores the path between two solutions in the search space. In this context, a path between a pair of solutions s_1 and s_2 explores other solutions that share attributes of these solutions. To generate the desired paths, begin with an initial solution s_1 and select moves that gradually include attributes coming from the guide solution s_2 .

According to [Ribeiro and Resende, 2012], good solutions to combinatorial optimization problems have a significant number of attributes in common. Furthermore, the basic premise of PR, according to these same authors, is that unknown high-quality solutions can be uncovered by investigating pathways connecting previously established good solutions.

3.4. Iterated Greedy

The IG is a metaheuristic classified as Stochastic Local Search. Given an initial solution, IG partially destroys the solution and then, using constructive heuristics, reconstructs the candidate solution [Ruiz and Stützle, 2007]. Based on this premise, IG can be divided into three stages: (i) components of a current solution S are removed, generating a partial solution S_p ; (ii) from S_p , a new complete candidate solution S' is constructed; and (iii) an acceptance criterion decides whether this process continues from S or S' .

An advantage of using IG is the fact that its implementation is simple, as the reconstruction process can reuse existing constructive heuristics, reducing the number of parameters to be defined. This approach is presenting high-quality solutions to several problems in the literature, such as Flow Shop Scheduling [Pisinger and Ropke, 2007; Yuan et al., 2008] and Vehicle Routing Problems [Ruiz and Stützle, 2007].

4. The Proposed Approach

We propose an IG with PR and RVND (IG-PR) for CAP, as a way to evolve the best generated heuristic by the hyper-heuristic in [Correa et al., 2022], establishing a well-defined constructive heuristic, criteria and parameterizations consistent with the application, as well as the incorporation of PR as an additional search resource.

The IG-PR is presented in Algorithm 1, where the input parameters are the problem instance (I), the maximum reconstruction size (Δ), the freedom from randomness of the RGA (α), the number of iterations required to execute PR (β), the probability of running a test that reverses the sides of the corridor (γ), and k , used to define the number of IG iterations per size.

Algorithm 1: The Proposed IG-PR

Input: $I, \Delta, \alpha, \beta, \gamma, k$
Output: S^2

```

1  $S \leftarrow \text{construct}(I, \alpha);$ 
2  $S^1 \leftarrow \text{rvnd}(S, I);$ 
3  $(Pb, Br) \leftarrow \text{buildProbabilities}(S^1);$ 
4  $\delta = \Delta; i = 0;$ 
5 while  $\delta > 0$  do
6    $S^2 \leftarrow \text{reconstruct}(S^1, \delta, Pb, Br);$ 
7    $S^3 \leftarrow \text{rvnd}(S^2);$ 
8   if  $i$  is divisible by  $\beta$  then  $S^3 \leftarrow \text{pathRelinking}(S^3);$ 
9   if a probability of  $\gamma$  is reached then  $S^3 \leftarrow \text{testTheInversion}(S^3);$ 
10  if lowest cost is achieved then
11     $S \leftarrow S^3; i = 0; \delta = \Delta;$ 
12     $(Pb, Br) \leftarrow \text{buildProbabilities}(S);$ 
13  else
14     $\text{incrementIteration}(i, \delta, Br, \Delta, k);$ 
15  end
16 end
17 return  $S;$ 

```

A solution S is initially generated by the constructive heuristic (Line 1), followed by a local search through RVND (Line 2). From the local optimum obtained, the search follows with the IG-PR iterations. In line 3, the calculation and distribution of probabilities between facilities (Pb) are performed. The facility that presents the greatest cost reduction among the others is also returned (Br). In line 4, the number of facilities to be reconstructed (δ) and the counter of reconstruction iterations (i) are respectively defined. The loop between lines 5 and 16 comprises the iterative execution of reconstructions. In line 6, the layout is reconstructed, where δ facilities are removed from the corridor, being inserted one at a time, using the construction heuristic presented in Section 4.3. Subsequently, the solution goes through a local search (line 7). Line 8 performs the search through the PR at each β iterations. In line 9, with probability γ , the solution has its sides inverted if the inversion improves the cost. Between lines 10 and 15, the algorithm checks if there is an improvement in the solution cost. In this case, the current solution is updated with the changed layout, i and δ values are restarted, and the probabilities are updated. If there is no improvement, i and δ are updated. Once δ has been decreased to zero, the stopping criterion is reached and the algorithm returns the solution S .

The reconstruction probability for each facility and the number of IG iterations are defined

strategically in order to improve the ability to reduce the cost of the solution. They are described in Sections 4.1 and 4.2. Also, the constructive heuristics, RVND, and PR are detailed in Sections 4.3, 4.4, and 4.5, respectively.

4.1. Probability Distribution

Before the IG iterations, an analysis is carried out on each facility of the layout, checking the cost of each one in every other position. It is then possible to identify facilities that, if reconstructed separately, can reduce the cost of the solution. This procedure has two purposes:

- Define the probability of each facility (Pb). These values are used when selecting the facilities to be reconstructed. The sample space is composed of all rooms in the layout, and the probability of each one is directly proportional to its cost reduction capacity. When this analysis concludes that no facility is capable of improving the cost of the objective function, these probabilities become uniformly distributed.
- Inform the facility that presents the largest cost reduction (Br). No facility is returned when no facility is able to improve the solution. This information allows for an assertive improvement in the cost of the layout when $\delta = 1$.

If a lower cost layout is obtained, it is kept by IG as the current candidate solution. Therefore, at this point, this probabilistic analysis must be recalculated.

4.2. Number of IG Iterations

The algorithm limits the maximum number of facilities to be reconstructed ($0 < \delta \leq \Delta$). The number of IG iterations varies inversely proportional to δ , being defined as

$$maximum\ i = \begin{cases} 1, & \text{when } \delta = 1; \\ (\Delta - \delta) \times k, & \text{otherwise.} \end{cases} \quad (2)$$

where k is a user-defined parameter.

If $\delta = 1$, a single iteration is sufficient, once the probabilistic analysis can inform whether there is a facility in the layout capable of improving the solution. On the other hand, the proposal starts with a larger number of reconstructions ($\delta = \Delta$). This value is reduced while no improvement is observed until $\delta > 0$.

4.3. Constructive Approach

The constructive approach proposed in this work is based on the RGA presented in Section 3.1. It acts iteratively on a list of pairs of facilities (i, j) , where $i, j \in F$, ordered in decreasing order concerning c_{ij} . This ordering prioritizes the inclusion of pairs of facilities with the highest demand coefficient first. The list L includes pairs of facilities where at least one of the facilities is not in the solution. When a pair (i, j) is chosen, the facilities i and j that are not in the solution are attributed to the corridor. Each facility is allocated in the position that causes the smallest increase in the layout cost.

The constructive approach is presented in Algorithm 2, where I is the problem instance, α defines freedom from randomness of the RGA, and S is the problem solution. At the start of the algorithm (line 1), an empty solution S is created, and a list L of pairs of facilities is generated and sorted, respectively. At each iteration, a pair (i, j) is selected from L (line 3), and the facilities that compose it that are not in the solution are inserted (lines 4 and 5). The pairs of facilities that are in the solution are deleted from the list L during the update procedure (line 6).

Algorithm 2: Constructive Approach

Input: I, α
Output: S
1 $S \leftarrow \emptyset$; $L \leftarrow \text{createSortedListOfPairs}(I)$;
2 **while** facilities missing **do**
3 $(i, j) \leftarrow \text{getRandomPair}(\alpha, L)$;
4 **if** i is not allocated in S **then** $\text{allocate}(i, S)$;
5 **if** j is not allocated in S **then** $\text{allocate}(j, S)$;
6 $\text{update}(L)$
7 **end**
8 **return** S ;

4.4. Random Variable Neighborhood Descent

There are several neighborhood structures usually adopted for CAP in the literature. Four strategies were presented in [Correa et al., 2022] to generate different candidate solutions in the neighborhood of a given one. In this work, the proposed RVND considers these strategies, which can be defined as:

- **Neighbor swap:** Swaps every pair of neighboring facilities on the same side of the corridor;
- **Non-Neighbor swap:** Every pair of non-neighboring facilities on the same side of the corridor is swapped;
- **Opposite Side Swap:** Swaps every pair of facilities on opposite sides.
- **Insertion:** Every facility on one side of the layout is removed from its current position and inserted in every position on the other side.

Each exploration strategy has two stopping criteria: (i) in case of improvement in the cost of the layout, restarting the RVND; (ii) when no exchange can reduce the cost of the layout, proceed to the next strategy. RVND is terminated when no exploration strategy produces an improvement in the current solution.

4.5. Path Relinking

The developed PR uses the current corridor as the initial solution. The target solution is the current corridor with its sides inverted, as shown in Figure 3. As one can observe, this inversion does not cause significant changes in the relative distances between pairs of rooms. This characteristic indicates that the inverted corridor has a slightly different cost when compared to the original one. The gradual execution of this inversion increases the chances that even better solutions can be found during the search process.

5. Computational Experiments

The computational experiments were performed using a set of 89 instances of size $n \in \{9, \dots, 70\}$, where n is the number of facilities. The instances are described as follows: $S\alpha$, $\alpha \in \{9, 9H, 10, 11\}$ [Simmons, 1969]; $Am\beta$, $\beta \in \{12a, 12b, 13a, 13b, 15\}$ [Amaral, 2012]; $N\gamma.\delta$, $\gamma \in \{25, 30, 40\}$, $1 \leq \delta \leq 5$ [Anjos and Vannelli, 2008]; $Sko.\epsilon.\zeta$, $\epsilon \in \{42, 49, 56\}$ and $1 \leq \zeta \leq 5$ and $Sko.64.5$ [Ahonen et al., 2014]; $P\theta$, $1 \leq \theta \leq 6$ [Amaral, 2008]; $ste36.0\eta$, $1 \leq \eta \leq 6$, [Anjos and Yen, 2009]; $CAP.\iota.\kappa.\lambda$, $\iota = \{30, 60, 90\}$, $\kappa = \{30, 40, 50, 60\}$ and $1 \leq \lambda \leq 3$ [Ahonen et al., 2014]; and finally, $AKV.\mu.05$, $\mu \in \{60, 70\}$ [Anjos et al., 2005].

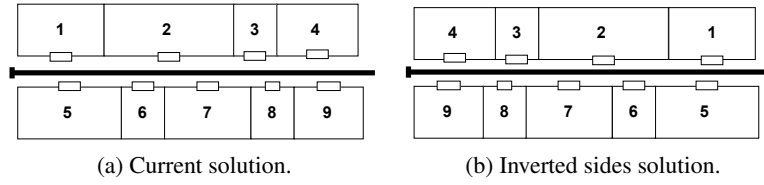


Figure 2: Example of inverting sides of a corridor in PR from (a) to (b).

The proposed approach was parameterized with $\alpha = 0.3$, $\Delta = 5$, $k = 70$, $\beta = 5$, and $\gamma = 0.1$, and the experiments were performed on a PC with an AMD Ryzen 5 2600 3.4GHz processor, 16.0GB DDR4, and Ubuntu 22.04 LTS. The proposal was implemented in C++11. The code, supplementary materials, and other resources are publicly accessible ¹.

A comparative analysis between the processors was necessary to mitigate their performance differences, equating them in order to enable a comparison of execution time. In [Herrán et al., 2021], an Intel Core i7 6500U 2.5GHz processor was used. According to the website CPU Benchmark², the AMD Ryzen 5 2600 processor presents a performance 24.4% superior to that of the Intel Core i7 6500U. In [Corrêa et al., 2024], the benchmark is equivalent. In this way, the comparative analyses are performed considering the equivalence of performances, converting the execution times based on the respective multiplicative constant.

The experiment was carried out in two parts. First, the proposed algorithm (IG-PR) was compared with the best approach proposed by the hyper-heuristic (GGPHH-IG) [Correa et al., 2022], which suggests the structure of an IG. The objective is to evaluate the quality of the IG-PR in relation to the GGPHH-IG against CAP. The second part consists of comparing the IG-PR with the best-known results in the literature, given by PBVNS [Herrán et al., 2021] as well as the best average execution time algorithm by CLONALG [Corrêa et al., 2024].

5.1. Analysis of the Results

In [Herrán et al., 2021], 10 independent runs were performed for instances $N\gamma.\delta$, $P\theta$ and $ste36.0\eta$. For other instances, 30 independent runs were performed. The cost of the best solutions for each instance and their average execution time were used in the comparisons. The same method was applied in [Correa et al., 2022; Corrêa et al., 2024]. In order to equate the analysis of results, this work uses the same execution parameters.

For the preliminary experiment, comparative analyses were made between GGPHH-IG and IG-PR, where the metrics used were the average time and cost gap according to results obtained in [Herrán et al., 2021]. The gap g of a given technique is defined as $g = (f - f^*)/f^*$, where f is the value obtained by the search approach and f^* is the best-known value from the literature. The gaps are considered here for the best cost of each instance. Table 1 shows these results. It can be observed that the IG-PR was able to largely outperform the GGPHH-IG, with an average gap of -0.0001% and considerably lower execution times.

The main experiment confronts the quality of the IG-PR developed with the PBVNS and CLONALG. Statistical analyses were performed considering the best cost obtained and the average execution time. The results are shown in Table 2 divided into two parts. The first part presents the result for instances with size $n \leq 49$. The names of the instances are presented in the first column. The second column contains the best costs and average times from the PBVNS. The third column presents these respective metrics reached by CLONALG. The fourth column shows the same results

¹<https://github.com/rafaelfreesz/CapIteratedGreedy>

²<https://www.cpubenchmark.net/>

Approach	Avg.	
	Time (s)	Cost Gap (%)
GGPHH-IG	47.8830	0.0915
IG-PR	11.5095	-0.0001

Table 1: Average gap for runtime and cost between GGPHH-IG and IG-PR against PBVNS.

obtained by the IG-PR developed in this work. The following columns present the second part for instances with size $n \geq 56$ in an analogous way. For each instance, the lowest average execution time and cost values are marked in bold. It is possible to observe that the three approaches were able to obtain the best cost up to instance P2. After that, CLONALG only achieved the best cost in 10 cases, PBVNS did so in 53 cases, and IG-PR finally achieved the best cost in 55 cases.

Instance	PBVNS		CLONALG		IG-PR		Instance	PBVNS		CLONALG		IG-PR	
	Cost	Time	Cost	Time	Cost	Time		Cost	Time	Cost	Time	Cost	Time
S9	1181.5	0.0900	1181.5	0.0063	1181.5	0.0058	Sko56.01	31972.0	7.4400	31972.0	5.5188	31972.0	5.6673
S9H	2294.5	0.1000	2294.5	0.0053	2294.5	0.0069	Sko56.02	248219.0	19.4600	8.8548	248216.0	14.7024	14.7024
S10	1374.5	0.1100	1374.5	0.0066	1374.5	0.0077	Sko56.03	85184.0	16.5900	85200.0	8.4947	85188.0	12.0603
S11	3439.5	0.1300	3439.5	0.0078	3439.5	0.0142	Sko56.04	156646.0	18.5100	156652.0	8.7758	156646.0	14.0232
Aml2a	1529.0	0.1500	1529.0	0.0122	1529.0	0.0170	Sko56.05	296176.5	19.8000	296229.5	9.7584	296195.5	15.4652
Aml2b	1609.5	0.1400	1609.5	0.0104	1609.5	0.0173	AKV.60.05	159643.0	28.2400	159704.0	11.3685	159642.0	18.1053
Aml3a	2467.5	0.1600	2467.5	0.0141	2467.5	0.0221	CAP.30.30.1	204089.0	27.2200	204105.0	13.1570	204089.0	22.6313
Aml3b	2870.0	0.1500	2870.0	0.0138	2870.0	0.0292	CAP.30.30.2	193199.5	26.3200	193266.5	12.3929	193177.5	19.1923
Aml5	3195.0	0.1900	3195.0	0.0282	3195.0	0.0391	CAP.30.30.3	161475.5	26.3800	161518.5	12.6383	161466.5	22.8024
N25.01	2302.0	2.1100	2302.0	0.1271	2302.0	0.2076	CAP.30.40.1	135133.5	27.4200	135192.5	13.1288	135133.5	19.8637
N25.02	18595.5	1.5290	18595.5	0.1816	18595.5	0.4359	CAP.30.40.2	159114.0	26.6900	159167.0	13.0228	159114.0	20.0116
N25.03	12114.0	1.5250	12114.0	0.1814	12114.0	0.4189	CAP.30.40.3	158967.5	26.8300	158999.5	13.2278	158951.5	22.5499
N25.04	24192.5	1.5600	24192.5	0.1942	24192.5	0.4546	CAP.30.50.1	110493.5	26.9400	110524.5	12.4286	110492.5	19.0560
N25.05	7819.0	1.4280	7819.0	0.1673	7819.0	0.2700	CAP.30.50.2	115302.0	25.9600	115376.0	12.6108	115296.0	22.6315
N30.01	4115.0	2.6000	4115.0	0.2926	4115.0	0.3852	CAP.30.50.3	114137.0	27.4400	114168.0	11.2323	114143.0	21.0872
N30.02	10779.5	2.5400	10779.5	0.4352	10779.5	0.5710	CAP.30.60.1	108117.0	25.4000	108132.0	13.7631	108113.0	18.3701
N30.03	22702.0	2.7900	22702.0	0.4410	22702.0	0.7949	CAP.30.60.2	109908.5	25.1100	109983.5	11.4008	109924.5	16.4776
N30.04	28401.5	2.9700	28401.5	0.5001	28401.5	0.7970	CAP.30.60.3	91619.0	24.8500	91685.0	12.1194	91642.0	17.3537
N30.05	57400.0	3.0100	57400.0	0.4631	57400.0	0.9535	CAP.60.30.1	445377.5	26.8700	445404.5	13.2897	445378.5	24.4248
P1	30282.5	2.8730	30282.5	0.6388	30282.5	0.9836	CAP.60.30.2	407890.5	26.5500	407902.5	12.1248	407896.5	21.1214
P2	33974.0	3.1550	33974.0	0.7210	33974.0	1.2634	CAP.60.30.3	416912.5	26.1500	416947.5	13.2324	416912.5	23.0645
P3	35052.5	3.1210	35054.5	0.7649	35052.5	1.8321	CAP.60.40.1	313304.0	25.7100	313343.0	12.6273	313310.0	21.5259
P4	34666.5	3.5720	34667.5	0.8071	34666.5	1.6606	CAP.60.40.2	320758.5	26.1100	320772.5	13.3171	320758.5	22.8922
P5	30771.0	3.2980	30771.0	0.7899	30771.0	1.5949	CAP.60.40.3	363020.5	25.4400	363071.5	13.4831	363020.5	23.7893
P6	34424.5	3.3770	34424.5	0.9147	34424.5	1.3134	CAP.60.50.1	273420.0	25.5500	273440.0	14.0089	273420.0	21.8376
ste36.01	4966.0	2.8050	4966.0	0.5929	4966.0	1.0219	CAP.60.50.2	269680.5	25.5800	269714.5	12.1258	269687.5	19.5394
ste36.02	87390.0	4.3290	87390.0	0.9838	87390.0	2.3165	CAP.60.50.3	295419.0	24.8500	295424.0	11.3689	295414.0	19.0735
ste36.03	50084.5	3.9700	50222.5	0.8750	50067.5	2.3509	CAP.60.60.1	227896.0	24.4900	227909.0	12.2961	227896.0	18.4422
ste36.04	46737.5	4.1910	46828.5	0.9477	46737.5	2.0630	CAP.60.60.2	246523.0	24.3800	246561.0	12.8266	246557.0	16.9133
ste36.05	44488.5	3.9550	44503.5	0.8462	44488.5	2.1757	CAP.60.60.3	206486.5	24.3100	206495.5	12.7837	206482.5	17.2783
N40.01	53722.5	5.1920	53730.5	1.8094	53724.5	2.8699	CAP.90.30.1	628850.0	25.8600	628859.0	14.1479	628850.0	28.9244
N40.02	48908.0	5.5560	48908.0	1.8702	48908.0	2.3319	CAP.90.30.2	561170.5	26.4600	561170.5	13.7308	561170.5	25.0543
N40.03	39250.5	5.4450	39250.5	1.7751	39250.5	2.6524	CAP.90.30.3	587816.5	26.7300	587816.5	14.3359	587816.5	23.6350
N40.04	38354.0	5.3700	38355.0	2.0169	38354.0	2.8385	CAP.90.40.1	474046.0	25.4900	474074.0	13.1721	474046.0	25.5471
N40.05	51496.0	5.4200	51500.0	1.7098	51496.0	3.1339	CAP.90.40.2	479963.0	25.7800	479976.0	13.5092	479963.0	27.0428
Sko42.01	12731.0	3.9000	12731.0	1.3755	12731.0	1.6224	CAP.90.40.3	512452.0	26.0900	512474.0	13.4921	512452.0	25.1386
Sko42.02	108006.5	7.2400	108024.5	2.5400	108006.5	4.2344	CAP.90.50.1	479683.0	24.8900	479706.0	12.8564	479683.0	22.8875
Sko42.03	86644.5	6.7100	86648.5	2.2090	86644.5	3.5440	CAP.90.50.2	445030.0	24.8900	445054.0	12.9279	445031.0	24.2095
Sko42.04	68701.0	6.5400	68726.0	2.0215	68701.0	3.7433	CAP.90.50.3	495052.5	24.9900	495068.5	13.2085	495052.5	19.6986
Sko42.05	124017.5	7.0500	124018.5	2.3230	124017.5	3.5272	CAP.90.60.1	385430.5	23.8100	385435.5	12.4334	385430.5	19.5205
Sko49.01	20470.0	5.1000	20470.0	2.8642	20470.0	3.7495	CAP.90.60.2	344775.0	24.4600	344776.0	13.4317	344775.0	22.7265
Sko49.02	208058.0	11.9000	208104.0	4.8189	208068.0	8.8904	CAP.90.60.3	411205.0	23.9500	411248.0	11.1041	411205.0	18.1795
Sko49.03	162182.0	11.6000	162207.0	4.5581	162180.0	7.8963	Sko64.05	250870.5	32.0600	250873.5	17.7105	250869.5	24.9065
Sko49.04	118246.5	11.1400	118263.5	4.7179	118246.5	7.0653	AKV.70.05	2109745.5	58.6500	2109900.5	22.3040	2109656.5	55.9395
Sko49.05	332836.0	12.1600	332879.0	4.4546	332811.0	6.8615	-	-	-	-	-	-	-

Table 2: Cost and runtime obtained compared to CLONALG and PBVNS.

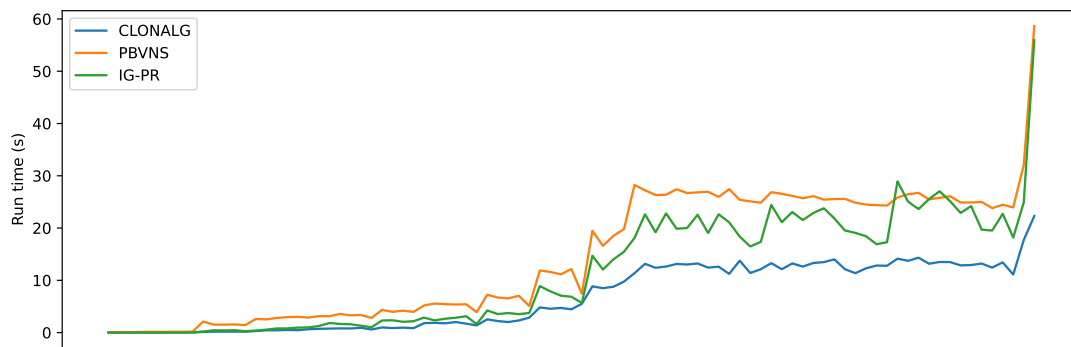
Table 3 presents summary statistics of the presented data. The first two rows count the quantities in which each heuristic was able to obtain the best average time and the best cost, respectively. The following rows present the average time and the overall average cost.

As can be seen, CLONALG was able to achieve consistently good time demands. However, it's possible to observe that the IG-PR algorithm was able to reduce the cost of the literature with a lower time demand than PBVNS, being able to achieve the best result in 76 instances.

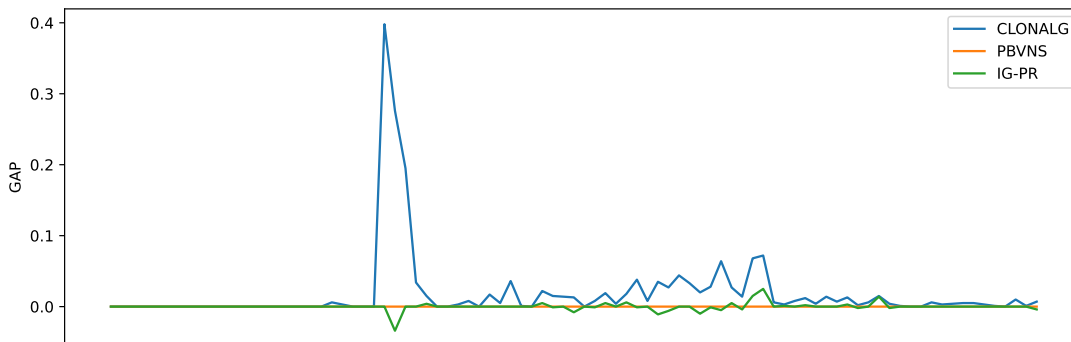
	PBVNS	CLONALG	IG-PR
# Best Time	0	88	1
# Best Cost	74	31	76
Avg. Time	14.59	6.84	11.51
Avg. Gap (%)	0.0000	0.0194	-0.0001

Table 3: General score and average runtime and cost for PBVNS, CLONALG and IG-PR.

For better visualization of the performance differences between IG-PR, PBVNS, and CLONALG, the graphic shown in Figure 3a demonstrates execution time and, Figure 3b shows the cost gaps between IG-PR and CLONALG against PBVNS according to instance size, in the same order as in Table 2. IG-PR presents a slightly smoother time growth curve compared to PBVNS, and although CLONALG presents better performance, the IG-PR gap curve is mostly lower, especially when faced with medium and large instances.



(a) Runtime versus instance size.



(b) Cost gap comparison of IG-PR and CLONALG against PBVNS.

Figure 3: Comparison Graphics

6. Concluding Remarks and Future Works

The positioning of rooms in a building under construction, machinery in production lines, and semiconductors in printed circuit boards are examples that affect the cost in industry and engi-

neering. The Corridor Allocation Problem (CAP), where facilities are placed along two rows with no empty space between the facilities and the first facility in each side must be at the beginning of the corridor, is solved here. We proposed an Iterated Greedy (IG) using Path Relinking (PR) and Random Variable Neighborhood Descent (RVND) as a local search technique (IG-PR).

The proposed IG-PR was developed based on a heuristic automatically generated by a Grammar-based Genetic Programming Hyper-heuristic (GGPHH). That heuristic presented iterative characteristics of partial destruction and reconstruction of the solution, followed by local search techniques, similar to an IG.

The proposed IG-PR was applied to 89 CAPs commonly used in the literature. The results obtained by IG-PR were statistically analyzed concerning the best results available in the state of the art [Herrán et al., 2021; Corrêa et al., 2024].

IG-PR improved the average execution times from [Herrán et al., 2021], and achieved better solution costs for 76 of the 89 instances. Finally, the average gap of IG-PR concerning the cost was -0.0001% while its average execution time was 11.51s.

We focused here on the best heuristic generated by a hyper-heuristic from the literature. However, other techniques can be explored as future work. Also, in [Herrán et al., 2021], the approach is a parallel search technique. Thus, a natural extension of the proposed IG-PR is to perform the search exploring high-performance computing. One may also notice that this improvement can use both CPU (as in the reference) and GPU. In addition, a different version of the PR adopted here can be developed considering other strategies. Finally, new local search strategies can be incorporated into the current RVND.

Acknowledgments

We thank the financial support provided by FAPEMIG, CAPES, CNPq, and UFJF.

References

- Ahonen, H., de Alvarenga, A. G., and Amaral, A. R. (2014). Simulated annealing and tabu search approaches for the corridor allocation problem. *European Journal of Operational Research*, 232(1):221–233.
- Amaral, A. R. (2008). An exact approach to the one-dimensional facility layout problem. *Operations Research*, 56(4):1026–1033.
- Amaral, A. R. (2012). The corridor allocation problem. *Computers & Operations Research*, 39(12):3325–3330.
- Anjos, M. F., Kennings, A., and Vannelli, A. (2005). A semidefinite optimization approach for the single-row layout problem with unequal dimensions. *Discrete Optimization*, 2(2):113–122.
- Anjos, M. F. and Vannelli, A. (2008). Computing globally optimal solutions for single-row layout problems using semidefinite programming and cutting planes. *INFORMS Journal on Computing*, 20(4):611–617.
- Anjos, M. F. and Yen, G. (2009). Provably near-optimal solutions for very large single-row facility layout problems. *Optimization Methods & Software*, 24(4-5):805–817.
- Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., and Qu, R. (2013). Hyper-heuristics: A survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724.

- Cellin, B. D. P. (2017). Métodos para resolução eficiente de problemas de layout. Master's thesis, Universidade Federal do Espírito Santo, Vitória.
- Chung, J. and Tanchoco, J. (2010). The double row layout problem. *International Journal of Production Research*, 48(3):709–727.
- Correa, R. F., Bernardino, H. S., de Freitas, J. M., Soares, S. S., Gonçalves, L. B., and Moreno, L. L. (2022). A grammar-based genetic programming hyper-heuristic for corridor allocation problem. In *Brazilian Conference on Intelligent Systems*, p. 504–519. Springer.
- Corrêa, R. F. R., SOARES, S. S. R. F., MORENO, L. L. O., GONÇALVES, L. B., BERNARDINO, H., and FREITAS, J. M. d. (2024). Clonal selection algorithm for corridor allocation problem.
- Duarte, A., Sánchez-Oro, J., Mladenović, N., and Todosijević, R. (2018). *Variable Neighborhood Descent*, chapter 3, p. 341–367. Springer International Publishing, Cham. ISBN 978-3-319-07124-4. URL https://doi.org/10.1007/978-3-319-07124-4_9.
- Gao, W., Friedrich, T., Neumann, F., and Hercher, C. (2018). Randomized greedy algorithms for covering problems. In *Proceedings of the Genetic and Evolutionary Computation Conference*, p. 309–315.
- Ghosh, D. and Kothari, R. (2012). Population heuristics for the corridor allocation problem. Technical report, Indian Inst. of Mgmt at Ahmedabad, Res. and Publication Dept.
- Glover, F. (1997). *Tabu Search and Adaptive Memory Programming — Advances, Applications and Challenges*, chapter 1, p. 1–75. Springer US, Boston, MA. ISBN 978-1-4615-4102-8. URL https://doi.org/10.1007/978-1-4615-4102-8_1.
- Herrán, A., Colmenar, J. M., and Duarte, A. (2021). An efficient variable neighborhood search for the space-free multi-row facility layout problem. *European Journal of Operational Research*, 295(3):893–907.
- Pisinger, D. and Ropke, S. (2007). A general heuristic for vehicle routing problems. *Computers & operations research*, 34(8):2403–2435.
- Ribeiro, C. C. and Resende, M. G. (2012). Path-relinking intensification methods for stochastic local search algorithms. *Journal of heuristics*, 18:193–214.
- Ruiz, R. and Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European journal of operational research*, 177(3):2033–2049.
- Santos, V. L. A. and Arroyo, J. E. C. (2017). Iterated greedy with random variable neighborhood descent for scheduling jobs on parallel machines with deterioration effect. *Electronic Notes in Discrete Mathematics*, 58:55–62. ISSN 1571-0653. URL <https://www.sciencedirect.com/science/article/pii/S1571065317300446>. 4th International Conference on Variable Neighborhood Search.
- Simmons, D. M. (1969). One-dimensional space allocation: an ordering algorithm. *Operations Research*, 17(5):812–826.
- Yuan, Z., Fügenshuh, A., Homfeld, H., Balaprakash, P., Stützle, T., and Schoch, M. (2008). Iterated greedy algorithms for a real-world cyclic train scheduling problem. In *Hybrid Metaheuristics: 5th International Workshop, HM 2008, Málaga, Spain, October 8-9, 2008. Proceedings 5*, p. 102–116. Springer.