

Clonal Selection Algorithm for Corridor Allocation Problem

Rafael F. R. Corrêa, Luciana B. Gonçalves, Stênio S. R. F. Soares,
Lorenza L.O. Moreno, Heder S. Bernardino, João M. Freitas

Universidade Federal de Juiz de Fora (UFJF)

Juiz de Fora, Brazil

{rafaelfreesz, lbrugiolo, ssoares,
lorenza, heder, joao}@ice.ufjf.br

ABSTRACT

Layout problems are combinatorial optimization problems that deal with the physical arrangement of facilities in a given area. In real-world situations, this type of problem can be applied to the efficient distribution of rooms in corridors, components on printed circuit boards, and machines in production lines. The Corridor Allocation Problem (CAP) is a typical layout problem where facilities are allocated in a two-sided corridor, not allowing empty spaces between them. Although exact methods have been presented, this problem is commonly addressed through approximate methods, given its complexity. Evolutionary algorithms, such as the Clonal Selection Algorithm (CLONALG), are widely used in combinatorial optimization problems. Thus, we propose here a CLONALG approach combined with Random Variable Neighborhood Search (RVND) for solving CAP. A set of 89 instances of the problem with up to 70 facilities was solved. The proposal achieved results statistically equivalent to those from the literature with a significant improvement in the processing times.

KEYWORDS. Corridor Allocation Problem, Clonal Selection Algorithm, Layout Problem, Evolutionary Algorithm.

AD&GP, AdP&ED, OA

1. Introduction

Layout problems deal with the physical arrangement of facilities in a given place Correa et al. [2022]. The distribution of rooms in buildings, production line machinery and semiconductors on printed circuit boards are important examples of applications for this type of problem Amaral [2012]; Correa et al. [2022]. The way these elements are positioned directly affects the functioning of the layout, where an appropriate distribution can represent increased efficiency and productivity, as well as reduced costs and space demand. As a result, an optimized layout can represent an increase in competitiveness for organizations and enterprises.

Concerning this type of problem, the Double-Row Facility Layout Problem (DRFLP) deals with the arrangement of facilities along two rows Chung and Tanchoco [2010], as illustrated in Figure 1. The Corridor Allocation Problem (CAP) was presented as a variation of DRFLP with two additional constraints Amaral [2012]: (i) empty spaces between facilities are not allowed, and (ii) the first facilities on each side start with abscissa zero.

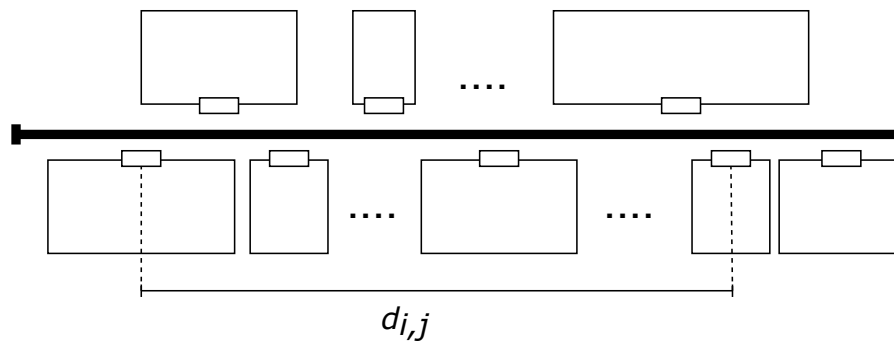


Figura 1: Illustrative example of an DRFLP problem. The distance between the pair of facilities i and j is defined by $d_{i,j}$.

Layout problems are NP-Hard Herrán et al. [2021], making the use of exact algorithms unattractive due to their high processing time demands Amaral [2012]. The use of meta-heuristics is appropriate for this type of problem. Although there is no guarantee of obtaining the best solution, meta-heuristics can reach good solutions in a short time, making their practical application feasible Talbi [2009].

Evolutionary Algorithms (EA) represent an important subset of the meta-heuristics. These algorithms are based on the principles of the natural evolution of species Talbi [2009]. In Ghosh and Kothari [2012], an EA was proposed for solving CAP. In addition, it is not hard to find in the literature the combination of EAs with a local search approach. Such a strategy aims at improving the convergence of EAs. Particularly, a Genetic Algorithm (GA) with Local Search (LS) and also Scatter Search (SS) with Path Relinking (PR) was proposed in Ghosh and Kothari [2012].

One can observe that other approaches can be used for solving CAP. In especial, Artificial Immune Systems (AIS) represents a set of EAs that emerged from the modeling of elements of immunological engineering. Specifically, the Clonal Selection Algorithm (CLONALG) de Castro and Zuben [2000]; de Castro and Von Zuben [2002] is based on the principle of clonal selection, combined with the process of affinity maturation, and adaptive immune response. CLONALG is commonly applied to optimization, machine learning, and pattern recognition problems. Examples of optimization problems that make use of CLONALG are Luo and Lin [2017]: the traveling salesman problem, vehicle routing problems, and scheduling problems. It is not hard to find works in the literature where CLONALG obtained good results when compared to GAs and other metaheuristics de Castro and Von Zuben [2002]; Bernardino and Barbosa [2009].

To the best of our knowledge, there is no work in the literature where CLONALG is adapted for solving CAP. Thus, we propose here a CLONALG approach to solve CAP. In addition, the proposal combines CLONALG with Random Variable Neighborhood Search (RVND) as a local search. Computational experiments were performed using 89 instances of the problem. In general, the proposed CLONALG with RVND reached results statistically similar to those from the literature with less processing time.

The remaining of this work is as follows. Section 2 presents some related work. In Section 3 the CAP and the methods used are formally presented. The developed algorithm is presented in Section 4. In Section 5, the computational experiments performed are described, as well as the statistical analyses. Finally, Section 6 concludes this work.

2. Related Work

Several papers related to CAP approach the problem using exact methods and metaheuristics.

The proposal initially made by Amaral [2012] uses Mixed-integer Programming (MIP), 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, since the problem is NP-Hard.

Two evolutionary algorithms were presented by Ghosh and Kothari [2012]. The first refers to a Genetic Algorithm allied to a local search. In the second approach, a Scatter Search with Path Relinking was presented. 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 and Tabu Search were used by Ahonen et al. [2014]. In this work, 68 instances of size $9 \leq n \leq 70$ were used, presenting better quality results than the previous work, with reduced time.

The most recent work in the literature presents a Parallel Basic Variable Neighborhood Search (PBVNS) Herrán et al. [2021], using a total of 89 instances of size $9 \leq n \leq 70$. The presented results are best present in the literature in a reduced execution time.

3. Materials and Methods

In this section, the methods used in the development of the proposal are presented. Section 3.1 presents CAP, the optimization problem solved. CLONALG is described in Section 3.2. Finally, Section 3.3 presents the RVND used here.

3.1. Corridor Allocation Problem

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 2. 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); and
3. Empty spaces are not allowed in consecutive pairs of facilities.

The abscissa of a facility $x_i, \forall i \in F$, 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 $d_{i,j} = |x_i - x_j|, \forall i, j \in F$.

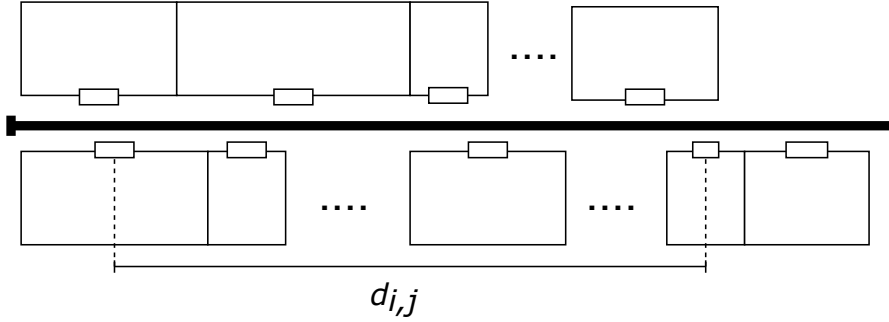


Figura 2: Illustrative example of a CAP. As in Figure 1, the distance between the pair of facilities i and j is defined by $d_{i,j}$.

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

$$\min \sum_{i \in F} \sum_{j \in F} f_{ij} \times d_{ij}, j > i. \quad (1)$$

A candidate solution to the problem consists of a permutation of the facilities.

3.2. Clonal Selection Algorithm

CLONALG is an evolutionary meta-heuristic present in the AIS suite. Its modeling was based on the clonal selection principle, whose characteristics allow organisms to develop an adaptive immune response when stimulated by antigens. In this principle, lymphocytes that react to antigens proliferate. For simplification, lymphocytes are not considered in this artificial scenario; antibodies are the candidate solutions and can proliferate (be cloned). In this way, antibodies are classified according to their antigenic affinities De Castro and Von Zuben [2000].

This algorithm has two versions: the first, developed for pattern recognition problems, and the second, for optimization problems. Here, we are interested in the latter.

The algorithm is composed of a population of antibodies Ab , where each individual/antibody has an affinity value $f_i, \forall i \in Ab$, a subset $Ab_n \in Ab$ of the individuals with the highest affinity in the population, a population C of N_c clones generated from Ab , a population C^* of matured clones, and a set $Ab_d \in Ab$, composed of the antibodies of lower Ab affinity. CLONALG operates according to the following steps:

1. **Generation of the Population (Ab):** Generate an initial population of antibodies and sort them according to their affinities (f);
2. **Cloning Procedure (C):** Reproduce the Ab_n antibodies, where the number of clones of each individual is proportional to its affinity, and can be defined as

$$N_c = \sum_{i=1}^N \text{round}(\beta \cdot N/i), 0 < \beta \leq 1, \quad (2)$$

where $N = |Ab|$, i is the index of the antibody in Ab , and β is a multiplicative user-defined parameter;

3. **Maturation Procedure (C^*):** Submit the population of clones (C) to a hypermutation process, where the mutation rate is inversely proportional to the affinity: the higher the affinity, the lower the mutation rate. At the end of this procedure, a population of mature antibodies is generated;
4. **Re-selection:** The best k antibodies from C^* are selected to compose Ab ;
5. **Regeneration (Ab_d):** Replace the d worst antibodies of Ab with new randomly generated individuals, as a way to maintain diversity; and
6. Repeat steps 2 through 5 until a stopping criterion is met.

3.3. Random Variable Neighborhood Descent

RVND is a local search heuristic derived from Variable Neighborhood Descent (VND) Duarte et al. [2018]. Different local search strategies allow the exploration of different search spaces. The purpose of VND is to perform an arbitrary number of local search approaches in a predefined sequence. In this way, the solution is able to reach local optima from the perspective of different strategies, allowing the exploration of broader regions in the search space. The difference of RVND is that the sequence in which the searches are performed is randomly defined in each run Santos and Arroyo [2017].

There are several movements in the literature. Regarding CAP, Correa et al. [2022] presented three strategies to generate new candidate solutions in the neighbor of a given one. These movements are adopted here and can be defined as:

- **Neighbor swap:** performs a swap of 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 are swapped; and
- **Opposite Side Swap:** every pair of facilities present on opposite sides of the corridor are swapped.

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 is able to reduce the cost of the layout, proceed to the next strategy. The RVND is terminated when no exploration strategy is able to produce an improvement in the objective function.

4. The Proposed Approach

The technique proposed here is based on CLONALG (Section 3.2). Also, the proposal combines CLONALG with an RVND (Section 3.3), which is used as a local search.

Here, the candidate solutions of CAP are represented by a vector of integers L of size n and an index t , such that $L[i]$, $0 \leq i < t$ represents the set of facilities positioned on the left side of the corridor, sorted from the beginning to the end. The range $L[i]$, $t \leq i < n$ represents the facilities distributed on the right side of the corridor, following the same ordering idea.

The representation of the solution with a vector of integers tends to be simple. This structure also facilitates permutation movements and cost calculations, which can be performed directly on the same data structure with a good performance.

As previously commented, the proposal is based on CLONALG. Algorithm 1 presents an overview of the processes involved throughout its execution. The input of the algorithm are defined by the user. In line 1, the population of size N is randomly initialized, adjusting the corridor division

index t so that the length of one side is not greater than the other in more than one facility. Lines 2-8 comprise the generations to be evolved throughout the process. In line 3, the cloning of the antibodies is carried out. The maturation process occurs in line 4. Subsequently, in line 5, RVND is applied to the matured clone with the highest affinity, followed by re-selection in line 6. Line 7 discards the worst antibodies in the population, generating new individuals in a process similar to line 1. At the end of the generations, in line 9, an RVND is performed on all antibodies in Ab_n . Finally, on line 10, the best individual in the population is returned.

Algorithm 1: CLONALG for CAP

Input: $N, gen, \alpha, \beta, \gamma$
Output: $Ab[0]$
 1 $Ab \leftarrow BuildInicialPopulation(N)$
 2 **for** $i = 1, \dots, gen$ **do**
 3 $C \leftarrow Reproduce(N, Ab, \alpha, \beta)$
 4 $C' \leftarrow Maturate(C, N, n)$
 5 $C'[0]' \leftarrow RVND(C'[0])$
 6 $Ab' \leftarrow Reselect(Ab, C')$
 7 $Ab'' \leftarrow Regenerate(Ab', \gamma)$
 8 **end**
 9 $Ab \leftarrow ImproveMemory(Ab, \alpha)$
 10 **return** $Ab[0]$;

As CAP is a minimization problem and, thus, the affinities of the antibodies are inversely proportional to the objective function value. The stopping criterion considered here is the number of generations (gen). The number of antibodies in Ab_n is defined based on a parameter α , as $|Ab_n| = \alpha * |Ab|$, $0 < \alpha \leq 1$. All C^* antibodies whose affinity is greater than at least one antibody in Ab are selected to be part of the population. The subset Ab_d is defined by the parameter γ , where $|Ab_d| = \gamma * |Ab|$, $0 < \gamma \leq 1$.

Algorithm 2: Reproduction operator

Input: N, Ab, α, β
Output: C
 1 $C \leftarrow \emptyset$
 2 $Ab_n = \alpha * |Ab|$
 3 **for** $i = 1, \dots, Ab_n$ **do**
 4 $N_c^i = (\beta * N) / i$
 5 **for** $j = 1, \dots, N_c^i$ **do**
 6 $C \leftarrow pushClone(Ab[i])$
 7 **end**
 8 **end**
 9 **return** C ;

The reproduction requires the number of clones and this is defined by Eq. (2). The cloning operator is presented in Algorithm 2. As input parameters, the algorithm receives the population, the number of antibodies in the population, α , and β . The population of clones is created in line 1. In line 2, the amount of antibodies to be cloned is calculated. Lines 3-8 scroll through Ab

until all antibodies from Ab_n have been cloned. Line 4 calculates the number of clones to be made, according to the index of the respective parent at Ab . In the loop of lines 5-7, clones of the individual are performed. In the end, the population of clones is returned in line 9.

The maturation process performs random swaps of facilities, where the number of swaps is randomly defined and is inversely proportional to the parent's index in the sorted Ab . The swaps are performed until a threshold value is reached. The maturation process is presented in Algorithm 3. The inputs are the clone population, population size, and instance size. Lines 1-7 loop through the list of cloned antibodies to perform sequences of mutations. In line 2, k receives the number of clones of the referred parent. In line 3, the proportional limit to be reached as a stopping criterion is defined. In the loop of lines 4-6, random exchanges of facilities are performed. Finally, on line 8, the population of clones is returned.

Algorithm 3: *Maturation operator*

Input: C, N, n
Output: C'
 1 **for** $i = 1, \dots, |C|$ **do**
 2 $k = \text{ClonesPerAb}(C[i])$
 3 $\text{goalProb} = 1.0 - (k/N)$
 4 **while** $\text{random}() \% 101 < \text{goalProb}$ **do**
 5 $C[i]' \leftarrow \text{swap}(\text{random}() \% n, \text{random}() \% n)$
 6 **end**
 7 **end**
 8 **return** C' ;

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].

CLONALG was run with a population of 10 individuals over 500 generations with the following parameterization: The α and γ parameters, shown in Section 4, were defined in 0.9 and 0.1, respectively. The β parameter, shown in Section 3.2, was defined in 0.1.

The experiments were performed on a PC with an AMD Ryzen 5 2600 3.4GHz processor, 16.0GB DDR4-1066GHz and Ubuntu 22.04 LTS. The algorithm was implemented in C++11 and is publicly accessible¹. The results were statistically compared to the Parallel Basic Variable Neighborhood Search (PBVNS), presented by Herrán et al. [2021], which uses a PC with an Intel Core i7 6500U 2.5GHz processor. According to the website www.cpubenchmark.net, the AMD Ryzen 5 2600 processor presents a performance superiority of 24.4% to the Intel Core i7 6500U. In this way, the comparative analyzes are made considering the equivalence of performances, converting the execution times based on the respective multiplicative constant.

Tables 1 and 2 show the costs and the average execution times obtained by CLONALG, as well as those obtained by PBVNS. The instances are identified in the first column. The second and

¹ github.com/rafaelfreesz/capClone.git

third columns contain the best cost and the average time obtained by the PBVNS, respectively. The fourth and fifth columns present the same metrics obtained by CLONALG. Finally, the last column presents the GAP between the solutions obtained by this work and those from Herrán et al. [2021]. Cells in bold highlight the best average time and best cost achieved for each instance.

Based on the two tables, it is possible to observe that CLONALG was able to reach zero gaps in 32 instances. Although in most cases this occurred for small and medium-sized instances, the algorithm was able to reach zero gaps in three large instances, such as CAP.90.30.2, CAP.90.30.3, and CAP.90.60.2, demonstrating conditions of achieving competitive results. Even in cases of higher gaps, given by instances ste.36.02, ste.36.03, ste.36.04, costs show little variation to the state of the art. In all instances of the problem, it is possible to observe a significant improvement in execution times.

Table 3 presents statistical analyzes containing average and standard deviation of gaps and execution times considering all instances used by both works. The results show that the average execution time of CLONALG was approximately halved, as well as its standard deviation. The average gap lower than its standard deviation indicates possible inefficiency of the algorithm in certain instances. However, even in the worst case, the gap value is small, demonstrating good performance.

Figure 3 shows the execution time (in seconds) for each instance for CLONALG and PBVNS. Instances are sorted in non-decreasing order, according to the number of facilities. It is possible to observe a significant improvement in CLONALG about the time demand, especially for large instances.

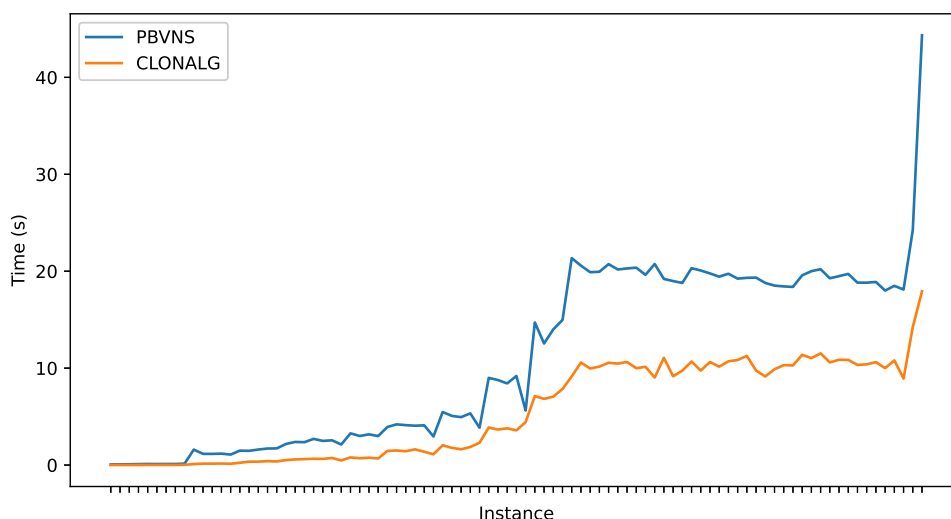


Figura 3: Execution time (in seconds) in relation to instances, in the same order they are presented in Tables 1 and 2.

6. Conclusions

Based on the practical applications of CAP, a good result must be allied to efficiency, being equally important to obtain it in the shortest possible time.

The use of CLONALG is promising because it allows the simultaneous exploration of a wide variety of solutions in the search space, a condition easily offered by evolutionary and po-

pulational algorithms. The perturbation provided by the hypermutation, allied to the local search provided by the RVND, allows the wide exploration of the search space with heterogeneous and refined solutions.

The results obtained show that the efficient use of CLONALG was able to provide results close to the state of the art, with a considerably shorter execution time, showing relevance when applied to CAP.

As a perspective for improving the presented approach, it is relevant for future work to consider other forms of local search, as well as the way to use it along with CLONALG. It is also important to dedicate efforts to defining the parameters used by the algorithm.

7. Acknowledgements

The authors thank the financial support provided by CNPq, Capes, FAPEMIG, and UFJF.

Referências

- 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.
- Bernardino, H. S. and Barbosa, H. J. C. (2009). *Artificial Immune Systems for Optimization*, p. 389–411. Springer, Berlin.
- 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.
- de Castro, L. N. and Zuben, F. J. V. (2000). The clonal selection algorithm with engineering applications. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00), Workshop on Artificial Immune Systems and Their Applications*, p. 36–37, Las Vegas, Nevada. Morgan Kaufmann.
- De Castro, L. N. and Von Zuben, F. J. (2000). The clonal selection algorithm with engineering applications. In *Proceedings of GECCO*, volume 2000, p. 36–39.

Tabela 1: Análise comparativa de resultados para instâncias de tamanho $9 \leq n \leq 56$.

Instances	PBVNS		CLONALG		Gap (%)
	Cost	Time (s)	Cost	Time (s)	
S9	1181.5	0.0680	1181.5	0.0051	0.000
S9H	2294.5	0.0756	2294.5	0.0043	0.000
s10	1374.5	0.0832	1374.5	0.0053	0.000
s11	3439.5	0.0983	3439.5	0.0063	0.000
Am12a	1529.0	0.1134	1529.0	0.0098	0.000
Am12b	1609.5	0.1058	1609.5	0.0084	0.000
Am13a	2467.5	0.1134	2467.5	0.0113	0.000
Am13b	2870.0	0.1134	2870.0	0.0111	0.000
Am15	3195.0	0.1436	3195.0	0.0227	0.000
N25.01	2302.0	1.5952	2302.0	0.1022	0.000
N25.02	18595.5	1.1559	18595.5	0.1460	0.000
N25.03	12114.0	1.1529	12114.0	0.1458	0.000
N25.04	24192.5	1.1794	24192.5	0.1561	0.000
N25.05	7819.0	1.0796	7819.0	0.1345	0.000
N30.01	4115.0	1.4886	4115.0	0.2352	0.000
N30.02	10779.5	1.4666	10779.5	0.3498	0.000
N30.03	22702.0	1.6012	22702.0	0.3545	0.000
N30.04	28401.5	1.7063	28401.5	0.4020	0.000
N30.05	57400.0	1.7184	57400.0	0.3723	0.000
P1	30282.5	2.1720	30282.5	0.5135	0.000
P2	33974.0	2.3852	33974.0	0.5796	0.000
P3	35052.5	2.3595	35054.5	0.6149	0.006
P4	34666.5	2.7004	34667.5	0.6488	0.003
P5	30771.0	2.4933	30771.0	0.6350	0.000
P6	34424.5	2.5530	34424.5	0.7353	0.000
ste36.01	4966.0	2.1206	4966.0	0.4766	0.000
ste36.02	87390.0	3.2727	87738.0	0.7908	0.398
ste36.03	50084.5	3.0013	50222.5	0.7034	0.276
ste36.04	46737.5	3.1684	46828.5	0.7618	0.195
ste36.05	44488.5	2.9900	44503.5	0.6802	0.034
N40.01	53722.5	3.9252	53730.5	1.4545	0.015
N40.02	48908.0	4.2003	48908.0	1.5034	0.000
N40.03	39250.5	4.1164	39250.5	1.4269	0.000
N40.04	38354.0	4.0597	38355.0	1.6213	0.003
N40.05	51496.0	4.0975	51500.0	1.3744	0.008
Sko42.01	12731.0	2.9484	12731.0	1.1057	0.000
Sko42.02	108006.5	5.4734	108024.5	2.0418	0.017
Sko42.03	86644.5	5.0728	86648.5	1.7757	0.005
Sko42.04	68701.0	4.9442	68726.0	1.6250	0.036
Sko42.05	124017.5	5.3298	124018.5	1.8674	0.001
Sko49.01	20470.0	3.8556	20470.0	2.3024	0.000
Sko49.02	208058.0	8.9964	208104.0	3.8737	0.022
Sko49.03	162182.0	8.7696	162207.0	3.6641	0.015
Sko49.04	118246.5	8.4218	118263.5	3.7925	0.014
Sko49.05	332836.0	9.1930	332879.0	3.5809	0.013
Sko56.01	31972.0	5.6246	31972.0	4.4363	0.000
Sko56.02	248219.0	14.7118	248239.0	7.1180	0.008
Sko56.03	85184.0	12.5420	85200.0	6.8285	0.019
Sko56.04	156646.0	13.9936	156652.0	7.0545	0.004
Sko56.05	296176.5	14.9688	296229.5	7.8444	0.018

Tabela 2: Análise comparativa de resultados para instâncias de tamanho $60 \leq n \leq 70$.

Instances	PBVNS		CLONALG		Gap (%)
	Cost	Time (s)	Cost	Time (s)	
AKV.60.05	159643.0	21.3494	159704.0	9.1387	0.038
CAP.30.30.1	204089.0	20.5783	204105.0	10.5764	0.008
CAP.30.30.2	193199.5	19.8979	193266.5	9.9621	0.035
CAP.30.30.3	161475.5	19.9433	161518.5	10.1594	0.027
CAP.30.40.1	135133.5	20.7295	135192.5	10.5537	0.044
CAP.30.40.2	159114.0	20.1776	159167.0	10.4685	0.033
CAP.30.40.3	158967.5	20.2835	158999.5	10.6333	0.020
CAP.30.50.1	110493.5	20.3666	110524.5	9.9908	0.028
CAP.30.50.2	115302.0	19.6258	115376.0	10.1373	0.064
CAP.30.50.3	114137.0	20.7446	114168.0	9.0292	0.027
CAP.30.60.1	108117.0	19.2024	108132.0	11.0636	0.014
CAP.30.60.2	109908.5	18.9832	109983.5	9.1646	0.068
CAP.30.60.3	91619.0	18.7866	91685.0	9.7423	0.072
CAP.60.30.1	445377.5	20.3137	445404.5	10.6830	0.006
CAP.60.30.2	407890.5	20.0718	407902.5	9.7466	0.003
CAP.60.30.3	416912.5	19.7694	416947.5	10.6370	0.008
CAP.60.40.1	313304.0	19.4368	313343.0	10.1506	0.012
CAP.60.40.2	320758.5	19.7392	320772.5	10.7051	0.004
CAP.60.40.3	363020.5	19.2326	363071.5	10.8385	0.014
CAP.60.50.1	273420.0	19.3158	273440.0	11.2612	0.007
CAP.60.50.2	269680.5	19.3385	269714.5	9.7474	0.013
CAP.60.50.3	295419.0	18.7866	295424.0	9.1390	0.002
CAP.60.60.1	227896.0	18.5144	227909.0	9.8843	0.006
CAP.60.60.2	246523.0	18.4313	246561.0	10.3108	0.015
CAP.60.60.3	206486.5	18.3784	206495.5	10.2763	0.004
CAP.90.30.1	628850.0	19.5728	628859.0	11.3729	0.001
CAP.90.30.2	561170.5	20.0038	561170.5	11.0376	0.000
CAP.90.30.3	587816.5	20.2079	587816.5	11.5240	0.000
CAP.90.40.1	474046.0	19.2704	474074.0	10.5885	0.006
CAP.90.40.2	479963.0	19.4897	479976.0	10.8595	0.003
CAP.90.40.3	512452.0	19.7240	512474.0	10.8457	0.004
CAP.90.50.1	479683.0	18.8168	479706.0	10.3347	0.005
CAP.90.50.2	445030.0	18.8168	445054.0	10.3922	0.005
CAP.90.50.3	495052.5	18.8924	495068.5	10.6178	0.003
CAP.90.60.1	385430.5	18.0004	385435.5	9.9947	0.001
CAP.90.60.2	344775.0	18.4918	344776.0	10.7972	0.000
CAP.90.60.3	411205.0	18.1062	411248.0	8.9261	0.010
Sko64.05	250870.5	24.2374	250873.5	14.2367	0.001
AKV.70.05	2109745.5	44.3394	2109900.5	17.9293	0.007

Tabela 3: Summarized score of the two works.

Score	PBVNS	CLONALG
Avg. Time (s)	11.0055	5.4985
Time SD (s)	9.0663	4.8558
Avg Gap (%)	0.000	0.019
Gap SD (%)	0.000	0.055

- de Castro, L. and Von Zuben, F. (2002). Learning and optimization using the clonal selection principle. *IEEE Transactions on Evolutionary Computation*, 6(3):239–251.
- Duarte, A., Sánchez-Oro, J., Mladenović, N., and Todosijević, R. (2018). *Variable Neighborhood Descent*, p. 341–367. Springer, Cham. ISBN 978-3-319-07124-4. URL https://doi.org/10.1007/978-3-319-07124-4_9.
- 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.
- 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.
- Luo, W. and Lin, X. (2017). Recent advances in clonal selection algorithms and applications. In *Symposium Series on Computational Intelligence (SSCI)*, p. 1–8, Honolulu, Hawaii, USA. IEEE.
- 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.
- Talbi, E.-G. (2009). *Metaheuristics: from design to implementation*. John Wiley & Sons.