Pesquisa Operacional (2022) 42: e245885 p.1-25
doi: 10.1590/0101-7438.2022.042.00245885

Printed version ISSN 0101-7438 / Online version ISSN 1678-5142
www.scielo.br/pope
ARTICLES

PROPOSITION OF A MATHEMATICAL PROGRAMMING MODEL FOR ALLOCATING HUMAN RESOURCES CONSIDERING MULTIPLE FACTORS AND USING DIFFERENT HEURISTICS

Ítalo Ruan Barbosa de Aquino[1], Josenildo Ferreira da Silva Junior[2], Maísa Mendonça Silva[3], Lúcio Camara e Silva[4*] and Ana Paula Cabral Seixas Costa[5]

Received November 24, 2020 / Accepted January 31, 2022

ABSTRACT. Human Resource Allocation (HRA) can be defined as the way professionals are distributed across the organization's tasks, given that each individual has his/her own set of characteristics, and that each task has specific needs. Thus, this paper puts forward a mathematical programming model for allocating human resources that considers employees' formal qualifications and experience and the possibility of employees sharing tasks in each project. The proposed mathematical model was designed and implemented according to a set of heuristics based on a Greedy Search (GS), a Genetic Algorithm, a Cosine Pigeon-Inspired Optimizer and an Iterated Local Search (ILS), to solve small, medium and large random instances. Thus, it was verified which of the heuristics had the best performance according to certain indicators, such as resolution time and average quality of the solutions found. Finally, they were also compared with the optimal solution obtained for small and medium-sized instances, with the best average results to ILS, although these are not too far from those of the GS.

Keywords: Human resource allocation, mathematical programming, resource sharing, resource qualification, Project Management

1 INTRODUCTION

Allocating human resources in an organization is one of the most difficult and daily problems for managers, as this is a combination of scheduling and decision-making problems involving the

*Corresponding author
[1]Universidade Federal de Pernambuco/ CAA, Av. Campina Grande, s/n, Km 59, Bairro Nova Caruaru, Caruaru, PE, Brazil – E-mail: italo_ruan_@hotmail.com – https://orcid.org/0000-0003-1610-8078
[2]Universidade Federal de Pernambuco/ CAA, Av. Campina Grande, s/n, Km 59, Bairro Nova Caruaru, Caruaru, PE, Brazil – E-mail: josenildo.junior9321@gmail.com – https://orcid.org/0000-0001-9411-038X
[3]Universidade Federal de Pernambuco/ Av. Prof. Moraes Rego, 1235, 50670-901 Recife, PE, Brazil – E-mail: maisa.ufpe@yahoo.com.br – https://orcid.org/0000-0003-4223-2769
[4]Universidade Federal de Pernambuco/ CAA, Av. Campina Grande, s/n, Km 59, Bairro Nova Caruaru, Caruaru, PE, Brazil – E-mail: luciocsilva@gmail.com – https://orcid.org/0000-0001-6043-0019
[5]Universidade Federal de Pernambuco, Av. Prof. Moraes Rego, 1235, 50670-901 Recife, PE, Brazil – E-mail: apcabral@hotmail.com – https://orcid.org/0000-0002-2932-4784

whole team and thus the more that members of the team are appropriately allocated to a particular project, the more efficiently, it will be developed (KHANIZAD & MONTEZER, 2018).

Human resources play a decisive role in the success of an organization, and managers, aware of this, look for the most efficient tools to optimize the use and allocation of their available resources across different services or systems, with the goal of maximizing or minimizing certain functions related to performance and productivity (BOUAJAJA & DRIDI, 2017).

However, this is not a simple task as there are many different combinations of possible employees and many often-conflicting factors to consider (such as time, cost, quality). Therefore, managing personnel selection and allocation without formal procedure can be very difficult and can lead the manager to forming a team that is not the best for a given situation (SILVA & COSTA, 2013).

Different approaches have been proposed in the literature to make the task of allocating resources more efficient (ARIAS et al., 2018). However, the different methods for human resource allocation generally do not take into account aspects such as interactions in the dynamic environment of the organization and expert knowledge (KHANIZAD & MONTEZER, 2018).

Therefore, this article puts forward a mathematical model for the problem of human resource allocation in projects, also considering the possibility of staff sharing and workers' know-how in each project (multiple aspects). The optimal allocation discussed here is related to minimizing the total execution time of all projects.

The application of the proposed model was performed on instances of different sizes, using Cbc Solver v.0.7.1, associated with the JuMP v0.21.6 package, from the Julia v1.6.1 language, for small and medium instances, as well as meta-based heuristics and a greedy heuristic for solving small, medium and large random instances implemented in the Julia language.

Two methods for solving combinatorial optimization problems which are well-known in the related literature were used: the Genetic Algorithm, which is widely applied to solve binary problems, and the Iterated Local Search (ILS). In addition, an easy-to-implement greedy heuristic was used to support the ILS because it provides faster computation. Moreover, the Cosine Pigeon-Inspired Optimizer meta-heuristic, formulated by Alazzam, Sharieh and Sabri (2020), was also used, as it presented competitive results with state-of-the-art algorithms to solve the binary problem addressed in this paper. Finally, for comparison purposes, the Cbc Solver, which uses Branch-and-cut, was applied to solve small and medium instances. It is worth stating that parameters were selected empirically in all resolution methods, except for explicit indications. Therefore, this paper maintained the original heuristics with no hybridization with GRASP (Greedy Randomized Adaptive Search Procedure). The proposed mathematical model was also applied to two test scenarios using the Cbc Solver to allocate human resources in an information system project.

This article is structured in 6 sections. After the introductory section, section 2 conceptualizes the resource allocation problem. In sections 3, 4 and 5 we present the mathematical modeling, the methodology of the proposed model and its application, respectively. Finally, in section 6

we make some final remarks, present some conclusions and make suggestions for future lines of research.

2 A BRIEF REVIEW OF THE LITERATURE ON RESOURCE ALLOCATION

In this section, the resource allocation problem is briefly conceptualized in subsection 2.1. Then, the problem of allocating human resources is fully described in subsection 2.2. Finally, some tools used in this context are presented in subsection 2.3.

2.1 Resource Allocation Problems

The resource allocation problem arises during the process of allocating resources across multiple projects or business units. This process seeks to find an optimal allocation of a limited amount of resources for a number of tasks. Resources can be people, assets, materials or capital that can be used to achieve a certain goal. The best solution can mean maximizing profits, minimizing costs or the best possible quality (LIN & GEN, 2008), or executing projects in the shortest possible time.

Osman et al. (2005) highlight some typical examples of resource allocation problems, as follows:

- Financing and investments: which involves allocating financial resources for different purposes (accounts receivable, inventory, etc.) over various time periods to maximize interest gains.
- Capital budgeting: which is related to the allocation of funds to projects that initially consume money but then generate revenue that maximizes a company's return on capital.
- Portfolio optimization: involves the allocation of funds to stocks or bonds to maximize return under a certain level of risk or to minimize risk under a desired rate of return.
- Job shop scheduling: This covers the allocation of time to work orders on different types of production equipment to minimize delivery time or maximize the use of equipment.

2.2 Human Resource Allocation

Human resource allocation aims to determine which task should be performed by whom (KANG et al., 2011). This is an area of research that has evolved over time, generating new proposals that are increasingly applied (ARIAS et al., 2018). This problem has been addressed in the literature in different applications, which include: software design (E SILVA & COSTA, 2013; KANG et al., 2011; PARK et al., 2015), in in health services (FAGERSTRÖM, 2009; LANZARONE & MATTA, 2014) and in a service industry (COROMINAS et al., 2006).

Many companies have been forced to incorporate how best to allocate human resources into their core strategy due to the dynamism of today's competitive environment. If allocated successfully, these can affect process performance, reduce costs and generate better productivity. Thus, the

adroit allocation of human resources can play a decisive role in the success of an organization (ARIAS et al., 2018; BOUAJAJA & DIDRI, 2017).

According to Bouajaja & Didri (2017), the problem of human resource allocation is a variation of the designation (or attribution) problem, which is a special type of linear programming problem in which the resources designated (machines, people, vehicles, etc.) are suitable for performing tasks; however, the number of assignees must be equal to the number of tasks because each assignee may be allocated to only one task. However, if problems do not fully match the model for an assignment problem (for example, if assignees can be assigned to more than one task or if a task needs to be performed by multiple assignees), the model must be redesigned (HILLIER & LIEBERMAN, 2010).

Many studies address the problem of allocating human resources. However, they do not consider some key factors that impact the applicability of assignments (CHIANG & LIN, 2020). In real cases, for example, the knowledge of experts in relation to a particular project or activity can be taken into account, and similarly it can be appropriate for people to share doing the same activities or projects, as presented in this paper.

Using matching models and graph theory, Numada (2021) developed a model to solve the problem of allocating human resources to assist in allocating people to emergency shelters following a disaster. The model seeks to reduce the number of hours worked to the minimum necessary, allowing for an appropriate rotation of people and a sufficient number of days off, taking into account a minimum number of people required per shelter each day. It is worth pointing out that the article does not explicitly consider budget limitations or sharing people between shelters.

Park et al. (2015) propose an approach to solve the problem of human resource allocation in software design by using a Genetic Algorithm (GA). In addition to minimizing the project duration, their approach takes into account practical issues such as the fact that a developer can work on more than one task at the same time, the assignment of activities that are related to each other, the size of the task, and the level of the team of developers. However, the skill level of each developer is not considered for each task specifically.

Chiang & Lin (2020) present a decision model for human resource allocation, dealing with a project divided into phases. However, it is not considered a scenario with several projects or simultaneous tasks that allows human resources to be shared. Liu et al. (2021) present a mathematical modeling for the human resource allocation problem in several university scientific research projects using an Improved Pigeon-Inspired Optimization (IPIO) algorithm. However, it is considered that researchers, after joining a project, cannot withdraw from the original project and join other projects before completing the tasks. Barreto et al. (2008) conduct an optimization approach that includes considering when an activity is small enough to be performed by a single developer.

In this article, the proposed model is intended to address the problem of human resource allocation in one or more projects, with the aim of minimizing the total time for completion. It is considered that the number of people allocated to each project influences the total time, that the

more expertise that the members of a project have between them, the better the project will be developed and that a person can work on more than one project, although there is a penalty related to multitasking. Finally, the model also considers a financial constraint related to including each person in each project.

2.3 Tools for Human Resource Allocation

Different methods of Operational Research are proposed in the literature to solve the problem of human resource allocation. To find an ideal solution, exact methods are used, including Branch and Bound techniques, dynamic programming, and the Hungarian method (BOUAJAJA & DIDRI, 2017).

The Hungarian method is an algorithm used to solve the classic assignment problem, where each assignee must perform a task. In the case where there is no balance between the number of assignees and tasks, assignees or ghost tasks are created so that this number is equal (HILLIER & LIEBERMAN, 2010). Branch and Bound has two main ingredients: branching and bounding. Branching refers to the way the search space of the solution is divided (tree-shaped), while bounding refers to the process of using valid boundaries to discard subsets of the search space without compromising optimization (PARRAGH & TRICOIRE, 2018). Dynamic programming is a useful mathematical technique for creating a sequence of interrelated decisions. It provides a procedure for determining a combination of optimal decisions (HILLIER & LIEBERMAN, 2010).

However, the human resource allocation problem is classified as an NP-hard combinatorial optimization problem (KHANIZAD & MONTEZER, 2018). Assigning tasks to employees in the context of production management is a difficult challenge mainly due to certain human factors and also because the allocation of human resources is associated with scheduling problems. In project management, such allocation is also difficult, as it is associated with time management and involves planning to ensure that resources are available when needed (BOUAJAJA & DIDRI, 2017).

Therefore, exact methods are not able to provide solutions for large instance problems within a reasonable time. Consequently, heuristic or metaheuristic methods are often chosen to solve real and practical problems and obtain a good, but not obligatory, optimal solution (BOUAJAJA & DIDRI, 2017).

To solve complex combinatorial optimization problems, metaheuristics are considered the most practical and efficient methods to provide good solutions in a reasonable computational time. These include Simulated Annealing (SA), Genetic Algorithm (GA), Tabu Search (TS), Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO) (BOUAJAJA & DIDRI, 2017).

Tchomté & Gourgand (2009) use an extension and adaptation of the PSO to solve the combinatorial optimization problem with precedence constraints in general and the project scheduling problem with resource constraints in particular. Deng et al. (2010) present a hybrid ant colony

optimization (HACO) approach to solving the problem of resource-constrained project planning. Park et al. (2015) use a GA to solve the problem of human resource allocation in a software project. Kang et al. (2011) optimize the scheduling of human resource allocations using a variation of SA. Al-Shourbaji and Zogaan (2021) propose the use of the Imperialist Competitive Algorithm to solve the human resource allocation problem in a cloud-based e-commerce service.

This paper uses the GA, the Cosine Pigeon-Inspired Optimizer (CPIO), the Iterative Local Search (ILS) and a greedy heuristic to solve the proposed problem. A GA is a meta-heuristic based on the Theory of Evolution proposed by Darwin, which is widely used in the context of Combinatorial Optimization (GOLDBARG et al., 2016). A GA has an initial population of individuals, who are subjected to an artificial process, similar to natural selection, dictated by genetic operators which will define how the selection of individuals will occur, their crossover and possible mutation of the generated individuals. The Pigeon-Inspired Optimizer (PIO) meta-heuristic, according to Alazzam, Sharieh and Sabri (2020), is a cloud intelligence heuristic based on the behavior of pigeons returning to their nests, which can be mimicked with a Map and by using Compass and Landmark operators. PIO is adapted to solve continuous variable problems, while CPIO is an adaptation of PIO to solve binary problems.

Finally, Lourenço, Martin and Stützle (2010) state that ILS is a method that, instead of searching through the entire space, seeks solutions based on solutions obtained from other methods, in general from a local search. This meta-heuristic can be summarized in 4 steps: creation of an initial solution, local search in an established neighborhood structure, perturbation of the current solution and setting and using criteria for accepting the solution.

3 PROPOSED MATHEMATICAL MODEL FOR HUMAN RESOURCE ALLOCATION

The mathematical model proposed in this article aims to define how best to allocate the available staff to projects, in order to minimize the total execution time.

This model also considers the possibility of sharing human resources between projects. This enables an improvement in the overall project execution time as more people may be involved in it. However, the attention of staff will be divided between more than one activity, so individual work on different projects will be done less efficiently than an exclusive dedication to a single project.

Expert knowledge is also assessed in the model. It is considered that an employee who specializes in a particular activity will perform his/her work more effectively, thereby reducing his/her completion time. On the other hand, lesser know-how will hinder making progress with the project.

Therefore, the model proposed in this article extends the proposal of Silva & Costa (2013) by considering two aspects: (i) the specific knowledge of people at the time of a project and (ii) the possibility of an employee being assigned to more than one project.

The model has the following decision variables:

- x_{ij}: binary variable that takes the value 1 if employee i is allocated to project j, and 0 otherwise.
- y_j: secondary variable that indicates the number of professionals allocated to the project j. This variable is obtained according to Equation (1), where NE refers to the total number of employees available and n to the total number of projects.

$$y_j = \sum_{i=1}^{NE} x_{ij},\ j = 1,2,\ldots,n \tag{1}$$

In addition to the NE and n parameters already shown, the model also uses the following parameters:

- c_{ij}: Costs of employee i in project j, i.e., how much the company pays for a given professional to work on a certain project.
- k_{ij}: time penalty (in months) in relation to the know-how of professional i in project j, i.e., the lower the know-how in a given project, the greater the value of k_{ij}, thus increasing the project completion time. On the other hand, the more skilled the professional is in a certain project, the lower the value of k_{ij}, can be until the null value, without any prejudice to the project.
- L: budget limit that the company has for human resources in all projects.
- $R_j(y_j)$: Refers to the time (in months) to complete project j using y_j professional. The use of this parameter is based on the study by E Silva & Costa (2013).
- t_i: time penalty (in months) for professional i, if involved in more than one project. If this occurs, the worker divides his or her attention into more than one project, and this leads to a drop in employee efficiency.

The proposed mathematical model is shown below:

$$minZ = \sum_{j=1}^{n} R_j(y_j) + \sum_{i=1}^{NE}\left[\left(\sum_{j=1}^{n} x_{ij} - 1\right) * t_i\right] + \sum_{i=1}^{NE}\sum_{j=1}^{n} x_{ij} * k_{ij} \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} \geq 1,\ i = 1,2,\ldots,NE \tag{3}$$

$$y_j \geq 1,\ j = 1,2,\ldots,n \tag{4}$$

$$\sum_{i}^{NF}\sum_{j}^{n} c_{ij}x_{ij} \leq L \tag{5}$$

$$x_{ij} \in \{0,1\},\ i = 1,2,\ldots,NE; j = 1,2,\ldots,n \tag{6}$$

Equation (2) is the Objective Function to be minimized, which is divided into three portions. The first refers to the time of completion of all projects by using a certain number of professionals in each project. The second portion deals with the penalty for sharing resources between projects. In this portion, the term $\sum_{j=1}^{n} x_{ij}$ evaluates how many projects employee i is allocated to. If he/she is allocated to only one project, the expression results in zero for this employee and there is no time penalty as this employee will be dedicated exclusively to one project. If he/she is allocated to two projects, for example, there will be a t_i penalty, if allocated to three, the penalty will be $2\,t_i$ and so on. This will be evaluated for each employee i, so there is also the sum in i. The third and last portion refers to the time penalty in terms of know-how, i.e., the lower the professional know-how in a given project, the greater the time penalty.

Constraints (3) ensure that each professional is allocated to at least one project. Inequality occurs due to the possibility of sharing human resources. Constraints (4) ensure that at least one person must be allocated to each project. Constraint (5) prevents staff costs on projects from exceeding the budget limit L that the company has for human resources on all projects. Finally, Constraints (6) deal with the nature of the variables.

4 METHODOLOGY

In this section, adaptations in the proposed human resources allocation model for the use of the Cbc Solver are described, in addition to addressing the meta-heuristics used, namely a Genetic Algorithm (GA), an Iterated Local Search (ILS), a Cosine Pigeon-Inspired Optimizer (CPIO), and a Greedy Search (GS). It is worth mentioning that methods for adjusting parameters were not used to make the algorithms more effective, parameters being selected empirically in all resolution methods, except when explicitly indicated.

4.1 Adaptations to the Model

To carry out the computational tests, Cbc Solver was used, an Open-Source Solver that uses Branch-and-Cut to solve mixed integer programming problems. Because the function $R_j(y_j)$ is possibly non-linear, it was necessary to replace it with equation (7) in the objective function (2) in order to solve the problem with a linear method.

$$\sum_{j=1}^{n}\sum_{i=1}^{NE} z_{ij} * ft_{ij} \tag{7}$$

$$y_j \geq i * z_{ij},\ j = 1,2,\ldots,n \tag{8}$$

$$\sum_{i=1}^{NE} z_{ij} = 1,\ j = 1,2,\ldots,n \tag{9}$$

$$z_{ij} \in \{0,1\},\ i = 1,2,\ldots,NE; j = 1,2,\ldots,n \tag{10}$$

The constants ft_{ij} were introduced, which are equivalent to the project duration j when i employees are allocated. In addition, binary variables z_{ij} were added, which assume a value of 1 if i employees are allocated to project j, and 0 otherwise. This requirement is defined in constraints (8). Constraints (9) oblige exactly one option for the number of employees to be chosen, and constraints (10) are relative to the nature of the variables z_{ij}.

4.2 Solution and Fitness Function

All heuristics handle only the x_{ij}, variables, assuming them as they are, an array of binary variables. Thus, it is possible that non-viable solutions may arise because of constraints (3), (4) and (5). As the meta-heuristics used here, in their basic versions, are not adapted to deal with restrictions, a penalty was used in all of them based on the number of violated restrictions, and also on the percentage of violation for the restriction case (5) of costs.

$$P = \sum_{j=1}^{n} R_j(1) \tag{11}$$

$$p_3 = \sum_{i=1}^{NE} max\left[1 - \sum_{j=1}^{n} x_{ij}, 0\right] \tag{12}$$

$$p_4 = \sum_{j=1}^{n} max[1 - y_j, 0] \tag{13}$$

$$p_5 = \left(1.0 + \frac{\left(\sum_i^{NF} \sum_j^n c_{ij} x_{ij} - L\right)}{L}\right) * u\left(\sum_i^{NF} \sum_j^n c_{ij} x_{ij} - L\right) \tag{14}$$

$$F = Z + P * (p_3 + p_4 + p_5) \tag{15}$$

Where u is the unit step function:

$$u(y) = \left\{ \begin{matrix} 1, y > 0 \\ 0, y \leq 0 \end{matrix} \right\} \tag{16}$$

In equation (11), the P value prevents the heuristics from considering non-feasible solutions better than the viable solutions. However, P can restrict the search space, especially because it was defined empirically, and not according to a minimum penalty norm addressed by Coello (2002).

Equations (12) and (13) calculate how many restrictions of groups (3) and (4), respectively, were violated. Equation (14), in turn, in addition to returning at least 1.0 when the solution is violated, this being verified with the help of the step function of equation (16), calculates the violated percentage as a way to differentiate small from large variations in the budget. Finally, these equations are united, together with Z (2), in (15), in the fitness function F, common to all meta-heuristics, and the result of equations (12), (13) and (14) is multiplied by P to penalize unfeasible solutions.

4.3 Genetic Algorithm

The Genetic Algorithm (GA), illustrated in algorithm 1, consists of selection, crossover, mutation and update operators, which are applied in a population of solutions, in order to mimic natural selection.

Algorithm 1 Genetic Algorithm.

```
Initialize population
while termination condition has not been met do
    Select parents
    Crossover
    Mutate
    Renew population
end while
```

To initialize the population, Reeves (2010) recommends the generation of M random solutions, with M defined by equation (17) for binary problems:

$$P_2^* = \left(1 - (1/2)^{M-1}\right)^l \tag{17}$$

$P_2^* = 99.9$, is assumed herein, as discussed by the author, where l is the number of binary variables of the instance to solve.

For the problem of the proposed model, the chosen operators were:

- Tournament Selection. In Select, n individuals from the population are chosen at random to participate in a tournament, and the most suitable tournament participant is selected. In the problem under analysis, n is equal to 20% of the size of the population.
- Two Point Crossover. In Crossover, the Two Point Crossover is undertaken, i.e., two points in the interval i = 1,2, ..., NE, are generated randomly: from the beginning of the solution to the first point and from the second point to the end of the solution, the son inherits genetic information from father 1 for all projects, while, among the points, the child inherits from father 2, generating one son. In the construction of the solution algorithm, it was determined that the number of children generated is the same as the population size.
- Bitwise mutation. All of the individual's genes are susceptible to mutation, with a probability p of occurring, occurring in Mutate. For the model in question, $p = 1/l$, being one of the ways addressed by Reeves (2010).

The population is updated in *Renew* according to the elitist technique, keeping 60% of the fittest individuals of the father generation and 40% of the fittest children. The population is restarted exactly one time if a better solution is not found after 20 generation, with 40% of the fittest individuals being retained.

Therefore, because of the effort to maintain population diversification, and because parameter adjustment methods are not used to define the maximum number of generations, the established stopping rule was based on the fitness function (REEVES, 2010). In this article, the GA will stop if the number of generations without finding a better solution is equal to 40.

4.4 Cosine Pigeon-Inspired Optimizer (CPIO)

The CPIO meta-heuristic mimics the behavior of pigeons returning to their nests using the Map and Compass and Landmark operators, illustrated in algorithm 2.

Algorithm 2 Cosine Pigeon-Inspired Optimizer (CPIO).

1: Receive Nc1, Nc2
2: Initialize pigeons
3: MapAndCompass (pigeons, Nc1)
4: Landmark (pigeons, Nc2)

Initially, this procedure generates the set of solutions, or pigeons, at random, as described by Alazzam, Sharieh and Sabri (2020). In the present paper, it was established that there are M pigeons, with the same M representing the GA population size.

In MapAndCompass, in each iteration, in a total of Nc1 iterations, the X_g position of the best pigeon is searched, so that the speed v_p of each pigeon p can be calculated using equation (18), and then one proceeds to update the X_p position of each pigeon with equation (20). The sigmoid function, shown in equation (19), is used to calculate the position.

$$v_p = Cosine\ Similarity(X_g, X_p) = \frac{X_g * X_p}{||X_g|| * ||X_p||} \tag{18}$$

$$S(v_p(t)) = \frac{1}{1 + e^{(\frac{-v_p}{2})}} \tag{19}$$

$$X(t)_{(i,p)}[i] = \left\{ \begin{matrix} X(t-1)_p[i], se\,(S(v_p(t)) > r) \\ X(t-1)_g[i], c.c. \end{matrix} \right\} \tag{20}$$

Where t is the current iteration, t-1 the past iteration, i is the index in the position of X_p to be changed, or not, and r a generated value of uniform distribution [0, 1).

At this stage, there is the possibility of changing repeated solutions in 20% of their current state, according to Alazzam, Sharieh and Sabri (2020). Therefore, in addition to searching for solutions equal to the end of each MapAndCompass iteration, a random number with a uniform distribution [0, 1) will be generated and, if it is greater than 0.5, the repeated solutions will be modified.

Finally, the Landmark operator will be executed, which will halve the number of pigeons in each iteration, with this division being rounded in this paper. Thereafter, X_c, corresponding to the central position, will be calculated based on the remaining population according to equation

(21), being duly rounded, and thus, all pigeons and their X_p positions will be updated according to equations (18), (19) and (20), with X_c as X_g.

$$X_c(t) = \left[\frac{\sum X_i(t) * F(X_i(t))}{N_p * \sum F(X_i(t))} \right] \tag{21}$$

Similarly to GA, Nc1 was changed to correspond to the number of iterations without improvements, being established 60 for this parameter. Nc2 was established for the Landmark operator to finish with 1 pigeon.

4.5 Greedy Search (GS)

The chosen GS, illustrated in algorithm 3, initializes the solution s with all values of x_{ij} i equal to 0. According to the fitness function F of equation (15), SearchForBestFitness searches for the best option among the os $x_{ij} = 0$ in s to receive value 1, returning the solution s* with exclusively this position changed in relation to s.

Thereafter, it is checked if s* is better than solution s. If so, the solution s will receive the values of s*, otherwise, the loop will be ended with the aid of the variable o.

Algorithm 3 Greedy Search.

```
1: Initialize s
2: o = 0
3: while o = 0 do
4:     s* ← SearchForBestFitness (s)
5:     if F (s*) < F(s) then
6:         s ← s*
7:     else
8:         o = 1
9:     end if
10: end while
```

4.6 Iterated Local Search (ILS)

The ILS meta-heuristic formally consists of steps of initial solution generation, local search, perturbation and acceptance criteria, illustrated in algorithm 4, below.

Initialize a solution s, using a neighborhood structure to search for the best solution s* from the neighborhood of s, usually in a LocalSearch process. In k_{max} iterations, at each iteration, aiming to distance the search from local optimal solutions, a Perturbation is applied to s, generating a new solution s', which will be submitted to LocalSearch to find s*', which in turn, will be evaluated to determine if it will become the best solution s* with the help of the AcceptanceCriterion. Finally, s will receive s*' for carrying out the perturbation on top of the best variable of the iteration.

Algorithm 4 Iterated Local Search.

```
Initialize s, k ← 0
s* = LocalSearch (s)
while k < k_max do
    s'=Perturbation (s)
    s*'=LocalSearch (s')
    s*=AcceptanceCriterion (s*, s*')
    s ←s*'
end while
```

For the initialization of the algorithm solution s, Lourenço, Martin and Stützle (2010) suggest that for shorter runs a greedy method be used to generate the initial solution. Therefore, the GS described above was chosen. In all executed instances, it was defined that kmax = 10. The One-Flip proposed by Lü, Glover and Hao (2009) is in the neighborhood structure used in LocalSearch(s), which defines as neighbors of s all solutions that have exactly one position of the solution matrix with a different value if compared with S. The AcceptancerCriterion chooses the best solution among the options. Finally, the Perturbation changes the solution at random in 20%.

5 APPLICATION OF THE PROPOSED MATHEMATICAL MODEL

In this section, the way that the computational experiments of heuristics and the exact method were carried out, as well as the obtained results and relevant discussions are presented. Two test instances based on Information Systems (IS) projects are also evaluated.

For the construction of heuristics, for the storage of instances and results, PoissonRandom.jl v0.4.0, StaticArrays.jl v1.0.1, Traceur.jl v0.3.1, Distributions.jl v0.24.12 and XLSX packages were used, while .jl v0.7.6. JuMP.jl v0.21.6 and Cbc.jl v0.7.1 were used together to solve the instances with Branch-and-Cut. Finally, the runtimes of the heuristics were recorded with BenchmarkTools.jl v0.5.0.

5.1 Evaluation of Heuristics

In this subsection the heuristics are evaluated based on a set of instances of different sizes.

In order to increase the execution speed of the algorithm, according to the tests in a larger instance analyzed in the article, in all algorithms, the solution is initially generated empty, and the possible total penalty is calculated and registered. At each modification in a position in this solution, the new fitness is registered based on the previous value as well as the difference that occurs in the solution when it is changed.

5.1.1 Creating Instances

Instances of different sizes were generated, with random values for the constants of the model, within uniform continuous distributions established in an intuitive way. For the cost constants, c_{ij},

$[300,700]$, for the time penalty in relation to know-how, k_{ij}, $[0,5]$, for the percentage r, $[0.4,0.7]$, which helps to define the spending limit at $L = r * \sum c_{ij}$, and for the time penalty t_i n relation to multitasking, $[0.5,3]$.

To generate the R_j function, initially a random value is generated within the interval $[15,60]$ that defines the project time if there is only 1 employee. From this, NE–1 s_{ij} values are generated distributed between $[0.01,0.08]$, for each j value. Thus, R_j can be calculated from i = 2, according to equation (22), but before s_{ij} are ordered, in relation to j, so that it has a greater impact to allocate employees to projects when they have a smaller number of employees already allocated.

$$R_j(i) = R_j(i-1) * (1 - s_{ij}) \tag{22}$$

The small instances generated were from the combination of number of employees and number of projects $NE = \{5,10,15,20\}$ and $n = \{3,5,10,15\}$, with an instance being generated for each of the 16 possible combinations. For medium-sized instances m, of the same, $NE = \{20,24,28,32\}$ and $n = \{18,22,26,30\}$. Finally, in large instances l, unlike the others, 15 instances of the combinations $NE = \{50,100,150,200\}$ and $n = \{20,30,40,50\}$, were generated, with the instance of size 200x50 not being generated because the package used to record instances does not support strings of such dimensions.

5.1.2 Computational results of heuristics

The experiments were carried out on a notebook with an i5-8265U processor and 8 GB of RAM memory. The heuristics were performed for all instances, whereas Cbc Solver was used for small and medium instances.

All solving methods covered in the article were executed 10 times for each instance, with the results found relating to the performance of the heuristics, as well as the exact method, in relation to time and fitness function recorded in Appendices A, B and C for the small, medium and large instances respectively.

Although GS is faster according to data in Appendix A, it is seen that time is not an issue for the heuristics used, while it is seen to scale the Cbc resolution time for larger problems.

To compare the methods in relation to the quality of the solutions, equation (23) was used, which measures the percentage difference PD between the performance of the optimal/best solution found BS, and the mean MS result obtained by the method evaluated in each instance. From this, the average percentage difference obtained by the method in the set of instances is calculated.

$$PD = \frac{MS - BS}{BS} * 100 \tag{23}$$

Thus, Table 1, Table 2 and Table 3 were generated, with the performance of the best heuristic for the instance highlighted in bold.

Table 1 – PD for small instances.

Instance	Problem Size (NE x n)	GS	GA	CPIO	ILS
s1	5x3	1,691%	0,451%	1,120%	0,165%
s2	5x5	1,188%	0,426%	2,160%	0,023%
s3	5x10	0,641%	1,165%	4,082%	0,271%
s4	5x15	0,410%	1,281%	7,547%	0,105%
s5	10x3	0,493%	3,052%	14,627%	0,148%
s6	10x5	0%	2,2980%	11,7890%	0%
s7	10x10	1,322%	1,871%	15,378%	0,586%
s8	10x15	0,740%	0,507%	13,630%	0,072%
s9	15x3	0,009%	10,585%	33,759%	0,004%
s10	15x5	0,194%	3,600%	19,412%	0,103%
s11	15x10	0,711%	3,392%	39,778%	0,538%
s12	15x15	0,989%	1,300%	31,932%	0,469%
s13	20x3	0,685%	7,629%	68,513%	0,577%
s14	20x5	0,191%	6,053%	41,109%	0,161%
s15	20x10	0,280%	2,744%	42,930%	0,159%
s16	20x15	0,922%	1,643%	42,449%	0,433%
	Mean	0,654%	3,000%	24,388%	0,238%

Table 2 – PD for midsize instances.

Instance	Problem Size (NE x n)	GS	GA	CPIO	ILS
m1	20x18	0,492%	1,335%	46,045%	0,323%
m2	20x22	0,712%	0,554%	47,469%	0,060%
m3	20x26	0,930%	0,798%	54,259%	0,149%
m4	20x30	0,756%	0,685%	61,818%	0,110%
m5	24x18	0,793%	1,464%	59,169%	0,165%
m6	24x22	1%	1,265%	61,380%	0,267%
m7	24x26	0,685%	0,711%	55,634%	0,153%
m8	24x30	0,620%	0,854%	120,365%	0,097%
m9	28x18	0,749%	2,557%	92,998%	0,588%
m10	28x22	0,383%	1,298%	85,593%	0,165%
m11	28x26	0,578%	1,245%	87,200%	0,235%
m12	28x30	0,711%	1,168%	91,195%	0,281%
m13	32x18	0,679%	2,948%	118,009%	0,541%
m14	32x22	0,582%	1,870%	107,803%	0,334%
m15	32x26	0,555%	1,288%	109,385%	0,272%
m16	32x30	0,456%	1,347%	108,772%	0,272%
	Mean	0,645%	1,337%	81,693%	0,251%

As it is possible to observe, among the formulated heuristics, the one with the highest performance is the ILS in all instances, and in most large instances, it is possible that it is not managing to leave the optimal location that is the initial solution of the method obtained by the GS. The second place is the GS, with an average performance close to the ILS, being followed by the GA and CPIO, which had a poor performance compared to the other methods.

Table 3 – PD for large instances.

Instance	Problem Size (NE x n)	GS	GA	CPIO	ILS
l1	50x20	0,026%	3,374%	190,996%	0,024%
l2	50x30	0,080%	2,368%	202,493%	0,052%
l3	50x40	0,401%	2,592%	240,737%	0,095%
l4	50x50	0,893%	1,129%	200,742%	0,144%
l5	100x20	0%	15,816%	674,395%	0%
l6	100x30	0%	7,882%	606,922%	0%
l7	100x40	0%	5,024%	565,278%	0%
l8	100x50	0,089%	3,995%	593,750%	0,079%
l9	150x20	0%	31,479%	1091,337%	0%
l10	150x30	0%	15,206%	1059,604%	0%
l11	150x40	0%	9,756%	996,708%	0%
l12	150x50	0%	7,023%	990,529%	0%
l13	200x20	0%	51,486%	1738,224%	0%
l14	200x30	0%	23,936%	1572,621%	0%
l15	200x40	0%	15,632%	1483,045%	0%
	Mean	0,099%	13,113%	813,825%	0,026%

The answers were obtained after running a certain number of generations in GA, iterations in CPIO or after visiting a specific number of neighborhoods in ILS, and these values are registered in Table 4 for all instances.

Table 4 – Average number of iterations in instances.

Num. Instances	GA			CPIO			ILS		
	s	m	l	S	M	l	s	m	l
1	54,3	377,4	877,7	91,7	78,2	80,7	35,7	586,5	1601,2
2	66,7	439,2	1324,2	99,8	81,2	81,8	48	725,2	2429,6
3	97,1	529,5	1601,5	117,8	82,7	80	83,3	843,5	3256,3
4	125	608,6	2173,8	78,2	84,9	81,8	120,8	980,1	4107,3
5	103	447,5	1572,4	83,1	92	90,4	53,7	701,4	3152
6	104,1	511,2	2322,1	106,6	78,9	85,5	79,1	867,1	4818,8
7	135,9	623,8	3190,3	105,8	84	84,3	160,8	1018,8	6451,1
8	186,9	686,7	3993,3	81,9	84,8	85,4	246,4	1182,9	8156,6
9	106,4	499	2084,1	68,8	79,6	87,5	74,8	808,6	4631,6
10	144,9	616,9	3455,3	78,3	88,9	89,6	121,3	999,7	7216
11	167,9	711	4535,2	75,3	87,1	93,4	235,4	1176,9	9704
12	272,1	800,5	5732,3	84,6	79,3	87,9	364,9	1360,6	12163,9
13	140,6	577,6	2808,9	76,7	89,5	92,4	99,6	937,4	6166,5
14	170,7	665	4372,4	75	82,6	91,7	154,3	1150,7	9507,4
15	274,5	784,2	5850,7	77	78,2	83,3	320	1359,2	12863,6
16	302,6	897,6	-	82,4	78,7	-	482,7	1556,8	-

Thus, it is clear that the amount relative to the number of iterations in each heuristic increases as the instance increases in size. However, this does not happen with the CPIO-based heuristic, which remains on average with the same number of iterations. This may indicate that the heuristic

cannot escape from optimal locations, which may be caused both by the penalty method and by the absence of a method to adjust the parameters.

5.2 Resolution of IS Test Instances

The proposed mathematical model was executed in two test scenarios using the Cbc Solver, because they are small scenarios.

Both scenarios assume that 10 employees need to be allocated to three projects. The times (in months) for project j completion using professional y_j, ($R_j(y_j)$), are shown in Table 5. These data were calculated and demonstrated in the study by E Silva & Costa (2013), to simulate IS project.

Table 5 – Time (in months) to complete projects based on number of professionals.

Number of employees	Project 1	Project 2	Project 3
1	21.1154	13.8237	8.2715
2	13.9778	9.1509	5.4755
3	8.6246	5.6463	3.3785
4	5.0558	3.3099	1.9805
5	3.5688	2.3364	1.398
6	2.3792	1.5576	0.932
7	1.487	0.9735	0.5825
8	0.8922	0.5841	0.3495
9	0.5948	0.3894	0.233
10	0.5948	0.3894	0.233

Source: E Silva & Costa (2013).

Table 6 shows the costs c_{ij} of employee i in project j. These were obtained conjecturally. These costs are related to how much the company pays for each professional to work on a certain project.

Table 6 – Costs of employee i in project j.

Employee	Project 1	Project 2	Project 3
1	1000	700	700
2	900	850	895
3	600	580	760
4	300	600	910
5	490	565	700
6	1000	900	900
7	780	640	645
8	820	810	880
9	900	950	930
10	1100	1000	900

Table 7 and Table 8 show, respectively, the time penalty k_{ij} in relation to the know-how of professional i in project j and the time penalty t_i in relation to professional i, if shared. These parameters were also conjectured.

Table 7 – Time penalty in relation to professional know-how in project j.

Employee	Project 1	Project 2	Project 3
1	0	0.2	0.2
2	0.1	0	0.15
3	0.3	0.3	0
4	0.11	0.11	0.11
5	0.2	0.17	0.05
6	0.03	0.13	0.09
7	0	0.05	0.04
8	0.2	0	0
9	0.14	0.12	0.06
10	0	0.02	0.05

The budget limit L, i.e., the financial capital that the company has available for human resources in all projects, was considered to be R$ 9,000.00.

Table 8 – Time penalty in relation to shared professional i.

Employee	1	2	3	4	5	6	7	8	9	10
t_i	1.5	1.0	1.1	2.0	1.7	0.9	1.2	1.3	1.0	1.4

In the first scenario, professionals were allocated without budgetary limitations. Although this situation is not common in reality, it can be seen to what extent staff sharing can be beneficial to the organization. In this case, the allocation of human resources occurs according to Table 9. The total execution time for this scenario was0.026 seconds.

Table 9 – Human resource allocation in the first scenario.

	Project 1	Project 2	Project 3
Allocated employees	1, 2, 4, 6, 7 and 10	2, 6, 8 and 9	3, 5, 6 and 9

It can be observed that there was sharing of employees: employee 2 (in projects 1 and 2), employee 6 (in all projects) and employee 9 (projects 2 and 3). The total execution time of the projects resulted in approximately 12.16 months. The total cost, if this solution were to be implemented, would be R$ 11,880.00.

Project 1 resulted in more staff allocated, six in all. Four employees were allocated to each of Projects 2 and 3.

In the second scenario, the financial limit is considered, assuming, consequently, a lower shared participation of human resources in the projects. The model was run for 0.310 seconds and the software found the results shown in Table 10.

Table 10 – Human resource allocation in the second scenario.

	Project 1	Project 2	Project 3
Allocated employees	1, 4, 6 and 10	2, 6, 7 and 8	3, 5 and 9

As expected, there was less resource sharing due to budget constraints: only employee 6 was shared (in projects 1 and 2). The total project execution time was longer in this case, resulting in approximately 13.07 months. The total cost of employees was R$ 8,990.

In the first scenario, the cost has become R$ 2,890.00 higher due to the greater performance of workers in the projects. Project execution, however, occurred 27.44 days faster. In the second case, the cost of human resources was reduced to within budget limits. However, projects take longer to complete.

Thus, it is found that greater resource sharing can be significant for better project execution performance in less time, despite the increase in the cost of human resources.

Thus, the model is able to allocate professionals between projects, also considering the possibility of sharing staff and taking into account the knowledge of workers in each project.

On the other hand, it is realistically very difficult to determine the time penalties for workers working on more than one project and also to quantify each other's know-how. Therefore, the model has limitations as to the accuracy in determining the parameters.

6 FINAL CONSIDERATIONS

Considering the improvements based on E Silva & Costa (2013), in this paper we proposed a mathematical model, to support the problem of human resource allocation.

In this model, the objective function to be minimized is divided into three portions. The first refers to the time of completion of all projects using a certain number of employees in each project. The second deals with the penalty for sharing resources between projects. In this case, if an employee is allocated to only one project, the expression results in zero for this employee and there is no time penalty as this employee will be dedicated exclusively to one project. If he/she is allocated to two projects, for example, there will be a penalty, and so on. The last portion refers to the time penalty in terms of know-how i.e., the lower the professional know-how in a given project, the greater the time penalty. Nonetheless, the model ensures that each employee is allocated to at least one project.

The model was then applied to several instances of different sizes, using the Cbc Solver and heuristics, among which, it was noted that the one with the best average performance is the ILS, with the CPIO heuristic having the worst performance. However, in tests with large instances, its performance does not differ much from the GS, and it is possible that the heuristic based on ILS cannot escape the great location that is the GS response.

Even so, the model was applied in two scenarios and budgetary limitations were either considered or ignored. Overall, it was noted that greater resource sharing can be significant for better project execution performance in less time, despite increased human resource costs.

It is worth noting that the way of penalizing solutions that violate restrictions was created in a simple and intuitive way. As a consequence of this, parts of the problem search space may become inaccessible to the heuristic, or, else, it can end with a solution that does not narrowly respect the restriction of expenses, which is one of the factors that may have harmed CPIO. Thus, the heuristic could have its performance improved by better defining the penalty function, or by using other techniques to deal with restrictions, such as those described by Coello (2002). Exploring these possibilities could be usefully undertaken in future studies.

In addition, to improve the performance of the proposed heuristic, another suggestion for future work is the use of methods to adjust the parameters of the heuristics, such as Revac, F-Race and SPO (MONTERO et al., 2014).

References

ARIAS M, SAAVEDRA R, MARQUES MR, MUNOZ-GAMA J & SEPÚLVEDA M. 2018. Human resource allocation in business process management and process mining: A systematic mapping study. *Management Decision*, **56**(2): 376–405.

ALAZZAM H, SHARIEH A & SABRI KE. 2020. A feature selection algorithm for intrusion detection system based on Pigeon Inspired Optimizer. *Expert Systems with Applications*, **148**.

AL-SHOURBAJI I & ZOGAAN W. 2021. A new method for human resource allocation in cloud-based e-commerce using a meta-heuristic algorithm. *Kybernetes*, ahead-of-print.

BARRETO A, BARROS M DE O & WERNER CML. 2008. Staffing a software project: A constraint satisfaction and optimization-based approach. *Computers & Operations Research*, **35**(10): 3073–3089.

BOUAJAJA S & DRIDI N. 2017. A survey on human resource allocation problem and its applications. *Operational Research*, **17**(2): 339–369.

CHIANG HY & LIN BMT. 2020. A decision model for human resource allocation in project management of software development. *IEEE Access*, **8**: 38073–38081.

COELLO CA. 2002. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer methods in applied mechanics and engineering*, **191**: 1245–1287.

COROMINAS A, PASTOR R & RODRÍGUEZ E. 2006. Rotational allocation of tasks to multi-functional workers in a service industry. *International Journal of Production Economics*, **103**(1): 3–9.

DENG L, LIN V & CHEN M. 2010. Hybrid ant colony optimization for the resource-constrained project scheduling problem. *Journal of Systems Engineering and Electronics*, **21**(1): 67–71.

E SILVA LC & COSTA APCS. 2013. Decision model for allocating human resources in information system projects. *International Journal of Project Management*, **31**(1): 100–108.

FAGERSTRÖM L. 2009.Evidence-based human resource management: a study of nurse leaders' resource allocation. *Journal of Nursing Management*, **17**(4): 415–425.

GOLDBARG MC, GOLDBARG EG & LUNA HPL. 2016. *Otimização Combinatória e Meta-Heurísticas: Algoritmos e Aplicações*. 1. ed. Rio de Janeiro: Elsevier.

HILLIER FS & LIEBERMAN GJ. 2010. *Introduction to Operational Research*. 8. Bookman.

KANG D, JUNG J, BAE D-H. 2011. Constraint-based human resource allocation in software projects. *Software: Practice and Experience*, **41**(5): 551–577.

KHANIZAD R & MONTAZER G. 2018. Optimal allocation of human resources based on operational performance of organizational units using fuzzy game theory. *Cogent Engineering*, **5**(1): 1466382.

LANZARONE E & MATTA A. 2014. Robust nurse-to-patient assignment in home care services to minimize overtimes under continuity of care. *Operations Research for Health Care*, **3**(2): 48–58.

LIN C-M & GEN M. 2008. Multi-criteria human resource allocation for solving multistage combinatorial optimization problems using multiobjective hybrid genetic algorithm. *Expert Systems with Applications*, **34**(4): 2480–2490.

LIU CB, MA, YH, YIN H & YU LA. 2021.Human resource allocation for multiple scientific research projects via improved pigeon-inspired optimization algorithm. *Science China Technological Sciences*, **64**(1): 139–147.

LOURENÇO HR, MARTIN OC & STÜTZLE T. 2010.Iterated Local Search: Framework and Applications. In: Gendereau M, Potvin J-Y. *Handbook of Metaheuristics*. New York: Springer, p.363–397.

LÜ Z, GLOVER F & HAO J-KJ. 2009. Neighborhood combination for unconstrained binary quadratic problems. *MIC 2009: The VIII Metaheuristics International Conference*, p. 1–7.

MONTERO E, RIFF MC & NEVEU B. 2014. A beginner's guide to tuning methods. *Applied Soft Computing*, **17**: 39–51.

NUMADA M. 2021.Development of matching modeling for human resource allocation of shelter management by the set theory. *Journal of Disaster Research*, **16**(4): 719–732.

OSMAN MS, ABO-SINNA MA & MOUSA AA. 2005. An effective genetic algorithm approach to multiobjective resource allocation problems (MORAPs). *Applied Mathematics and Computation*, **163**(2): 755–768.

PARK J, SEO D, HONG G, SHIN D, HWA J & BAE DH. 2015. Human resource allocation in software project with practical considerations. *International Journal of Software Engineering and Knowledge Engineering*, **25**(01): 5–26.

PARRAGH SN & TRICOIRE F. 2019. Branch-and-bound for bi-objective integer programming. *INFORMS Journal on Computing*.

REEVES CR. 2010. Genetic Algorithm. In: Gendereau M, Potvin J-Y. *Handbook of Metaheuristics*. New York: Springer, p.109–139.

TCHOMTÉ SK & GOURGAND M. 2009. Particle swarm optimization: A study of particle displacement for solving continuous and combinatorial optimization problems. *International Journal of Production Economics*, **121**(1): 57–67.

Internet

BENCHMARKTOOLS. *GitHub*. https://github.com/JuliaCI/BenchmarkTools.jl.

CBC. *GitHub*. https://github.com/jump-dev/Cbc.jl.

DISTRIBUTIONS. *GitHub*. https://github.com/JuliaStats/Distributions.jl.

JUMP. *GitHub*. https://github.com/jump-dev/JuMP.jl.

POISSONRANDOM. *GitHub*. https://github.com/SciML/PoissonRandom.jl.

STATICARRAYS. *GitHub*. https://github.com/JuliaArrays/StaticArrays.jl.

TRACEUR. *GitHub*. https://github.com/JunoLab/Traceur.jl.

XLSX. *GitHub*. https://github.com/felipenoris/XLSX.jl.

How to cite

AQUINO IRB, SILVA JUNIOR JF, SILVA MM, E SILVA LC & COSTA APCS. 2022. Proposition of a mathematical programming model for allocating human resources considering multiple factors and using different heuristics. *Pesquisa Operacional*, **42**: e245885. doi: 10.1590/0101-7438.2022.042.00245885.

A RESULTS OF HEURISTICS

A.1 Small instances

3 unfeasible solutions were registered as PIO outputs for instance l13.

Instance	Problem Size (NE x n)	Cbc		GS		GA				PIO				ILS			
		Optimal	Time (s)	Value	Time (s)	Best	Average	σ	Time (s)	Best	Average	σ	Time (s)	Best	Average	σ	Time (s)
l1	5x3	105,43	6,20E-03	107,21	1,79E-06	105,43	105,9	0,486	2,99E-04	105,43	106,61	1,456	1,24E-04	105,43	105,6	0,12	1,33E-05
l2	5x5	174,01	7,80E-03	176,08	4,26E-06	174,01	174,75	0,748	3,67E-04	174,74	177,77	1,74	1,59E-04	174,01	174,05	0,125	2,99E-05
l3	5x10	311,84	1,71E-02	313,84	1,18E-05	312,19	315,47	2,459	5,74E-04	319,52	324,57	4,037	2,49E-04	311,84	312,69	0,605	9,59E-05
l4	5x15	560,28	1,86E-02	562,57	2,68E-05	562,81	567,46	3,959	8,43E-04	588,38	602,56	8,62	3,57E-04	560,28	560,87	0,908	2,05E-04
l5	10x3	111,28	1,39E-02	111,83	6,54E-06	111,28	114,68	2,527	3,93E-04	118,3	127,56	5,313	1,69E-04	111,28	111,44	0,265	4,12E-05
l6	10x5	150,65	2,00E-02	150,65	1,24E-05	151,12	154,11	2,928	5,34E-04	160,44	168,41	6,96	2,38E-04	150,65	150,65	0	9,42E-05
l7	10x10	308,66	3,87E-02	312,74	4,16E-05	310,58	314,44	2,751	1,09E-03	335,6	356,13	13,061	4,39E-04	309,38	310,47	0,919	3,65E-04
l8	10x15	598,93	2,44E-02	603,36	9,24E-05	599,61	601,97	1,583	1,91E-03	658,89	680,57	11,004	6,08E-04	598,93	599,36	0,462	8,82E-04
l9	15x3	88,04	4,21E-02	88,04	1,39E-05	91,62	97,36	3,792	4,72E-04	106,43	117,76	7,843	2,29E-04	88,04	88,04	0,004	7,95E-05
l10	15x5	200,89	9,53E-02	201,28	2,86E-05	202,84	208,12	3,953	7,86E-04	233,71	239,88	5,889	3,35E-04	200,89	201,09	0,156	2,12E-04
l11	15x10	320,19	8,44E-02	322,47	7,57E-05	326,68	331,05	4,309	1,71E-03	420,73	447,56	12,63	6,45E-04	320,84	321,92	0,496	8,23E-04
l12	15x15	561,54	1,52E-01	567,09	1,55E-04	565,97	568,84	1,888	3,06E-03	714,23	740,85	19,118	9,19E-04	563,11	564,17	1,02	1,86E-03
l13	20x3	109,35	1,13E-01	110,1	2,58E-05	114	117,69	3,009	6,18E-04	133,02	184,27	65,471	2,64E-04	109,35	109,98	0,232	1,56E-04
l14	20x5	160,75	2,33E-01	161,06	5,39E-05	163,51	170,48	3,957	9,77E-04	218,54	226,84	7,09	4,05E-04	160,75	161,01	0,107	3,74E-04
l15	20x10	360,19	3,41E-01	361,2	1,64E-04	363,96	370,07	4,993	2,69E-03	496,58	514,82	11,765	8,33E-04	360,19	360,76	0,38	1,58E-03
l16	20x15	534,29	4,71E-01	539,21	3,56E-04	538,44	543,06	3,339	4,48E-03	727,54	761,08	21,085	1,25E-03	534,29	536,6	1,29	3,49E-03

A.2 Midsize instances

1 unfeasible solution was registered as PIO output for instances m4, m10 and m13, and 4 unfeasible solutions were registered for m8.

Instance	Problem Size (NE x n)	Cbc		GS		GA				PIO				ILS			
		Optimal	Time (s)	Value	Time (s)	Best	Average	σ	Time (s)	Best	Average	σ	Time (s)	Best	Average	σ	Time (s)
m1	20x18	637,44	6,97E-01	640,58	4,84E-04	642,9	645,95	2,237	5,80E-03	899,99	930,94	21,034	1,38E-03	638,4	639,5	0,696	5,02E-03
m2	20x22	841,12	6,59E-01	847,11	6,97E-04	843,83	845,78	1,874	7,91E-03	1181,05	1240,39	31,69	1,65E-03	841,21	841,63	0,348	7,50E-03
m3	20x26	910,43	8,10E-01	918,9	9,16E-04	914,13	917,7	3,038	1,01E-02	1354,63	1404,43	22,194	1,95E-03	910,67	911,79	0,511	1,07E-02
m4	20x30	1103,88	1,05E+00	1112,23	1,32E-03	1107,29	1111,44	2,953	1,33E-02	1593,07	1786,28	397,441	2,53E-03	1104,31	1105,09	0,806	1,39E-02
m5	24x18	662,37	1,37E+00	667,62	6,53E-04	667,22	672,06	2,885	7,59E-03	1018,69	1054,28	26,375	1,64E-03	663,04	663,46	0,339	7,22E-03
m6	24x22	809,91	2,46E+00	815,05	8,94E-04	816,83	820,15	4,217	9,75E-03	1255,85	1307,02	29,958	2,13E-03	810,78	812,07	0,916	1,07E-02
m7	24x26	1043,88	3,07E+00	1051,03	1,35E-03	1048,97	1051,3	2,356	1,39E-02	1572,35	1624,62	37,8	2,63E-03	1044,45	1045,47	0,58	1,54E-02
m8	24x30	961,23	3,57E+00	967,19	1,50E-03	965,7	969,45	2,467	1,76E-02	1642,03	2118,22	559,623	2,79E-03	961,45	962,16	0,406	1,99E-02
m9	28x18	563,25	4,92E+00	567,47	6,41E-04	571,49	577,66	4,531	9,02E-03	1027,43	1087,07	36,961	1,87E-03	565,11	566,57	0,902	9,45E-03
m10	28x22	807,21	7,95E+00	810,3	1,13E-03	813,79	817,69	3,411	1,32E-02	1345,99	1498,13	313,215	2,61E-03	807,6	808,55	0,86	1,46E-02
m11	28x26	916,57	6,64E+00	921,87	1,61E-03	923,81	927,98	2,97	1,78E-02	1586,84	1715,82	60,223	2,82E-03	917,31	918,72	0,709	2,06E-02
m12	28x30	1002,99	1,04E+01	1010,12	2,18E-03	1008,68	1014,7	3,692	2,19E-02	1877,97	1917,66	26,318	3,24E-03	1004,11	1005,8	1,351	2,76E-02
m13	32x18	552,96	1,36E+01	556,71	9,05E-04	560,15	569,26	4,34	1,23E-02	1068,66	1205,5	237,805	2,46E-03	554,54	555,95	0,726	1,26E-02
m14	32x22	687,63	2,05E+01	691,63	1,35E-03	695,05	700,49	4,37	1,63E-02	1382,61	1428,91	27,712	2,72E-03	688,45	689,92	0,972	1,92E-02
m15	32x26	882,2	1,42E+02	887,09	1,90E-03	889,93	893,56	2,348	2,01E-02	1784,15	1847,19	44,946	3,50E-03	883,52	884,59	1,034	2,78E-02
m16	32x30	999,26	7,56E+02	1003,81	2,55E-03	1007,88	1012,71	4,321	2,73E-02	2016,41	2086,17	47,336	3,66E-03	1001,56	1001,97	0,332	3,54E-02

A.3 Large instances

5 unfeasible solutions were registered as PIO outputs for instance l5, and 3 for instance l12, with the best performance found in the instance being highlighted in bold.

Instance	Problem Size (NE x n)	GS		GA				PIO				ILS			
		Value	Time (s)	Best	Average	Σ	Time (s)	Best	Average	Σ	Time (s)	Best	Average	Σ	Time (s)
l1	50x20	647,6	2,43E-03	662,84	669,27	4,96	2,74E-02	1815,54	1883,98	44,443	3,81E-03	**647,42**	647,58	0,054	3,73E-02
l2	50x30	958,61	4,80E-03	972,11	980,52	4,673	6,17E-02	2806,04	2897,39	57,749	6,23E-03	**957,84**	958,34	0,302	8,38E-02
l3	50x40	1162,91	7,68E-03	1169,38	1188,29	10,994	9,42E-02	3847,68	3946,63	52,358	9,27E-03	**1158,26**	1159,36	1,234	1,55E-01
l4	50x50	1668,86	1,40E-02	1664,92	1672,75	5,418	1,60E-01	4864,78	4974,52	70,664	1,13E-02	**1654,08**	1656,46	1,127	2,40E-01
l5	100x20	**494,43**	7,02E-03	548,78	572,63	11,841	8,82E-02	3327,82	3828,87	462,153	8,45E-03	**494,43**	**494,43**	0	1,46E-01
l6	100x30	**784,01**	1,33E-02	830,39	845,8	17,039	2,01E-01	5318,98	5542,33	132,748	1,46E-02	**784,01**	**784,01**	0	3,30E-01
l7	100x40	**1120,06**	2,44E-02	1163,28	1176,34	9,735	3,61E-01	7350,05	7451,53	96,372	1,85E-02	**1120,06**	**1120,06**	0	6,09E-01
l8	100x50	1387,39	4,00E-02	1430,23	1441,53	10,327	5,63E-01	9517,15	9616,45	72,689	2,40E-02	**1386,16**	1387,25	0,389	9,71E-01
l9	150x20	**464,39**	1,35E-02	589,93	610,57	19,047	1,78E-01	5170,81	5532,44	158,098	1,50E-02	**464,39**	**464,39**	0	3,24E-01
l10	150x30	**718,12**	2,68E-02	815,95	827,32	8,47	4,34E-01	8133,02	8327,38	114,815	2,34E-02	**718,12**	**718,12**	0	7,72E-01
l11	150x40	**1040,98**	4,73E-02	1122,07	1142,54	16,823	7,60E-01	10726,66	11416,5	262,881	2,84E-02	**1040,98**	**1040,98**	0	1,37E+00
l12	150x50	**1419,76**	6,75E-02	1503,75	1519,47	12,97	1,20E+00	14750,99	15482,93	1001,286	3,67E-02	**1419,76**	**1419,76**	0	2,19E+00
l13	200x20	**389,04**	2,21E-02	567,43	589,33	13,53	3,07E-01	6914,72	7151,34	105,042	1,79E-02	**389,04**	**389,04**	0	5,94E-01
l14	200x30	**685,35**	4,44E-02	815,26	849,39	18,468	7,44E-01	11253,41	11463,24	159,847	3,16E-02	**685,35**	**685,35**	0	1,38E+00
l15	200x40	**978,94**	7,75E-02	1103,8	1131,97	20,987	1,28E+00	15194,58	15497,11	149,538	4,21E-02	**978,94**	**978,94**	0	2,44E+00