CEFET/RJ
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO

HYBRID APPROACHES TO THE TWO-STAGE CAPACITATED FACILITY LOCATION PROBLEM

Igor da Silva Morais

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Ciência da Computação, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ, como parte dos requisitos necessários à obtenção do grau de mestre.

Orientadores:
Pedro Henrique González Silva
Eduardo Bezerra da Silva

Rio de Janeiro,
Setembro de 2022

Hybrid Approaches to the Two-Stage Capacitated Facility Location Problem

Dissertação de Mestrado em Ciência da Computação, Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, CEFET/RJ.

Igor da Silva Morais

Aprovada por:

Presidente, Prof. Pedro Henrique González Silva, D.Sc. (orientador)

Eduardo Bezerra da Silva, D.Sc. (coorientador)

Glaydston Mattos Ribeiro, D.Sc.

Vanessa de Almeida Guimarães, D.Sc.

Diego Nunes Brandão, D.Sc.

Rio de Janeiro,

Setembro de 2022

Ficha catalográfica elaborada pela Biblioteca Central do CEFET/RJ

M828 Morais, Igor da Silva
Hybrid approaches to the two-stage capacitated facility location problem / Igor da Silva Morais. — 2022.
42f. : il. (algumas color.) , enc.

Dissertação (Mestrado) Centro Federal de Educação Tecnológica Celso Suckow da Fonseca, 2022.
Bibliografia : f. 39-42
Orientador: Pedro Henrique González Silva
Coorientador: Eduardo Bezerra da Silva

1. Logística empresarial. 2. Produção enxuta. 3. Armazenamento. 4. Algoritmos. 5. Computação. I. Silva, Pedro Henrique González (Orient.). II. Silva, Eduardo Bezerra da (Coorient.). III. Título.

CDD 658.5

DEDICATÓRIA

DEDICATÓRIA Dedico essa dissertação a meu avô, Ivan Pontes Morais, que sempre me ensinou muito e deu todo suporte para que eu estudasse. Ele nos deixou, mas está comigo em todo o meu caminhar.

AGRADECIMENTOS

Agradeço a impermanência por me manter nesse ano tão difícil de aprendizado e construções. Agradeço também à minha esposa, Fernanda Fiorese Martins, que durante esse tempo cuidou do trabalho reprodutivo em nosso lar, permitindo assim, que eu pudesse realizar esse mestrado de forma mais tranquila. Agradeço aos meus pais e avós que me apoiaram e me deram suporte todo esse tempo. Agradeço em especial minha irmã, Laís da Silva Morais, que é uma inspiração para minha vida. Agradeço à minha sogra e cunhada pelo apoio, por acreditarem e por torcerem por mim. Agradeço aos meus orientadores, em especial ao Pedro Henrique González, pela paciência, pelos ensinamentos valiosíssimos, pelo entusiasmo e pela disponibilidade em me ajudar durante todo o processo. Agradeço também aos meus chefes, em especial ao Marcel Pedroso que é meu orientador de coração. Agradeço também à Vanessa de Almeida Guimarães que por muitas vezes conversou e me acalmou nos momentos de pressão. Por ultimo, mas não menos importante, agradeço meus amigos Bathazar Paixão, Gabriel Souto, Liss Faulhaber, Matheus SOBRENOME e Victor Mariano, com quem pude compartilhar momentos de ensinamentos.

RESUMO

Hybrid Approaches to the Two-Stage Capacitated Facility Location Problem

Igor da Silva Morais

Orientadores:
Pedro Henrique González Silva
Eduardo Bezerra da Silva

Resumo da Dissertação submetida ao Programa de Pós-graduação em Ciência da Computação do Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ como parte dos requisitos necessários à obtenção do grau de mestre.

Na classe de problemas de cadeia de suprimento, O problema de localização de facilidades em Dois Níveis (PFLC2n) consiste em encontrar localizações ótimas para instalação de fábricas e depósitos que atendam a demanda dos clientes. O objetivo do problema é minimizar os custos de abertura e fluxo, obedecendo as restrições de produção das fábricas, armazenamento dos depósitos e respeitando a demanda de clientes. Esse problema pode ser visto como aplicação do contexto de cidades inteligentes, pois cobre os três pilares: governança, energia e transporte. Para encontrar soluções para o problema duas hibridizações são propostas uma do Clustering Search(CS) com Adaptive Large Neighborhood Search(ALNS) e Local Branching, com o intuito de comparar e mostrar robustez dos componentes outra hibridização é implementada para um Biased Random-Key Genetica Algorithm (BRKGA com Local Branching chamado HBRKGA). Para comparação instâncias da literatura foram utilizadas. Os resultados mostram que para o PFLC2n a abordagem do HBRKGA supera o estado da arte atual para 44 de 50 instâncias e a estabilidade é mostrada por meio de uma análise estatística que testa a significância em comparação aos outros métodos.

Palavras-chave: Meta-heurística Híbrida; *Clustering Search*; *Biased Random-Key Genetic Algorithm*;*Local Branching*

Rio de Janeiro,

Setembro de 2022

ABSTRACT

Hybrid Approaches to the Two-Stage Capacitated Facility Location Problem

Igor da Silva Morais

Advisors:
Pedro Henrique González Silva
Eduardo Bezerra da Silva

Abstract of dissertation submitted to Programa de Pós-graduação em Ciência da Computação - Centro Federal de Educação Tecnológica Celso Suckow da Fonseca CEFET/RJ as partial fulfillment of the requirements for the degree of master.

In the class of supply chain problems, the Two-Stage Capacitated Facility Location (TSCFL) is defined by optimal locations for installing factories and warehouses to meet the demand of customers. The problem aims to minimize operating costs: opening facilities and the flow of products from factories to customers, passing through warehouses, meeting the capacity constraints of factories and warehouses and customers' demand. This problem can be viewed as a simplified version for application in the context of smart cities, as can cover all the three pillars,governance, energy and transportation. To solve this problem, two hybridization are proposed of Clustering Search (CS), Adaptive Large Neighborhood Search (ALNS) and Local Branching (LB) is proposed . This hybridization is a new and interesting approach which has found high quality solutions in low computational time.In order to compare and test robustness of the proposed components another hybridization is proposed a Hybrid approach of the Biased Random-key Genetic Algorithm. To show that, computational experiments were performed using benchmark instances. The results showed that the HBRKGA outperforms the current state-of-art for the TSCFL for 44 out of 50 instances and the stability of the methods is showed in a statistical analysis, in order to test differences of the method.

Key-words: Hybrid Meta-heuristic, Clustering Search, Biased Random-Key Genetic Algorithm and Local Branching

Rio de Janeiro,

Setembro de 2022

Contents

List of Figures

List of Tables

Lista de Abreviações

ALNS	Adaptive Large Neighborhood Search	14
BRKGA	Biased Random-Key Genetic Algorithm	14
CS	Clustering Search	14
GA	Genetic Algorithm	14
GRASP	Greedy Randomized Adaptive Search Procedure	14
HEA/FA	Hybrid Evolutionary Algorithm	14
LB	Local Branching	28
MIP	Mixed Integer Programming	17
RKGA	Random-key Genetic Algorithm	32
RVND	Random Variable Neighborhood Search	24
SGC	Semi-greedy Constructive	27
SGD	Semi-greedy Destructive	27
SGR	Semi-greedy Repairer	27
TSCFL	Two-Stage Capacitated Facility Location Problem	13

Chapter I Introduction

Success of any company is related to its operational efficiency, which is closely linked to the geographical layout of its facilities [Zhong et al., 2017]. A good location can provide a reduction in transportation costs, increasing the company's profits and meeting customers' demand efficiently.

The facility location problem proposes to define an optimal system for the distribution of facilities (factories, warehouses, points of sale, etc.) to meet customers' demand at the lowest possible cost. An extension of this problem is the Two-Stage Capacitated Facility Location Problem (TSCFL), which considers the capacity restrictions of factories and warehouses, and a level interposed between factories and customers (warehouses).

The products must flow from the factories through the warehouses until reaches the customers (factory $\rightarrow$ warehouse $\rightarrow$ customer). The final cost is defined by the cost of opening the facilities plus the cost to transport from factories to warehouses and from warehouses to customers. Therefore, minimizing this total cost is the objective of the problem.

In order to improve population quality of life we could think about the concept of smart cities as said in Guimarães et al. [2021]. And as a way of defining what is a smart city we could use the three pilars presented by governance, energy and transportation. The two stage facility location problem can be used as a way not only of improving transportation but can be viewed as way of reducing the energy costs, as the flow between the facilities will also be minimized. As a last structure for completing the three pilars are the governance, which can be improved in two ways companies can create monitoring structures to monitor changes in the flow and also public policies of transportation can be improve traffic which gives better quality of life without stopping the flow of production.

Variants of the TSCFL problem were proposed during the years. They concern either a single-source or multi-source with a single commodity or multi-commodity planning. Many combinations of this characteristics were studied in the literature. The single-source with multi-commodity was proposed by Geoffrion and Graves [1974], which used a Benders decomposition for solving it. Another variant, proposed by Tragantalerngsak et al. [1997], considers that the warehouses are supplied by only one factory. Pirkul and Vaidyanathan [1998] considered a formulation with multiple sources and multi-commodity and proposed a Lagrangian relaxation for solving it. Later, Mauri et al. [2021] proposed two metaheuristics for the same problem. Understanding that, from

a practical point of view, not only the warehouse location is important, but also its size, Wu et al. [2015] proposed a mathematical formulation that expands the original one and also considers the warehouse size. In addition, to deal with this variant, Wu et al. [2015] proposed a Lagrangian relaxation technique. Another variant which considers that the customers can only be attended by one warehouse is presented in Yang et al. [2019].

Another approaches of the TSCFL can be found in the literature, as seen in Klose [2000], Tragantalerngsak et al. [2000] and Litvinchev and Ozuna [2012]. Klose [2000] applied a Lagrangian relaxation based on *cut-and-relax* as an alternative for solving the problem, considering that the factories are fixed, which means that they are previously allocated. Fixed factories were also considered in Tragantalerngsak et al. [2000], which presented an exact method for solving the problem. Litvinchev and Ozuna [2012] has already proposed the use of Lagrangian relaxation to solve the problem considering the location of both factories and warehouses. More details on some of the different approaches to TSCFL can be found at Klose and Drexl [2005].

In this work, the same variant presented by Litvinchev and Ozuna [2012] is studied. This variant consists in determining the location of both warehouses and factories. This same approach was considered in the most recent works on TSCFL, described briefly below.

In Fernandes et al. [2014], a Genetic Algorithm (GA) was presented to solve the TSCFL. The authors proposed a set of instances with different characteristics to evaluate the proposed GA. This same set of instances was used in Louzada et al. [2016], where a hybrid method was proposed based on the combination of the Clustering Search (CS) method, to define the factories and warehouses to be installed, with an exact algorithm, used to define the flow of products between factories, warehouses and customers. The CS was able to find better solutions compared to the GA in lower computational times. A Hybrid Evolutionary Algorithm (HEA/FA) was presented in Guo et al. [2017]. The results obtained were competitive with the ones obtained with AG and CS described above. In Biajoli et al. [2019] a Biased Random-Key Genetic Algorithm (BRKGA) for the TSCFL was proposed. The results obtained were competitive with those presented so far and redefined the state-of-art for the TSCFL. Finally, in Gonzalez et al. [2019b], a hybrid method was proposed based on the Greedy Randomized Adaptive Search Procedure (GRASP) with a *Local Branching* procedure. The GRASP, presented in Gonzalez et al. [2019b], obtained relevant results and was able to keep the computational time stable in relation to the increase in the instances' size. This last work motivated the application of the local branch procedure used in this dissertation, as can be seen the stability in the amount of computational time, shows a relevant information for using it in this proposal.

To solve this variant of the TSCFL, we propose a hybridization of CS and Adaptive Large Neighborhood Search (ALNS) metaheuristics with the Local Branching technique to find solutions for the

TSCFL. A search made in Scopus, on January 12th 2022, using the following query string ("Clustering Search" OR "CS") AND (ALNS OR "ADAPTIVE LARGE NEIGHBORHOOD SEARCH") has presented three articles: 1) the product of this dissertation; 2) an article using ALNS and CS as separated components; 3) a Physics article. The result indicates that a hybridization between CS and ALNS has not been addressed in the literature yet. ALNS was proposed by Ropke and Pisinger [2006] and uses processes of destruction and reconstruction adaptively to improve the quality of the solution, while the CS [Oliveira et al., 2013] metaheuristic avoids excessive calls on local search procedures, which can reduce the computational time. After that, a Local Branching [Fischetti and Lodi, 2003] procedure is called, using the best solution found as base solution, in order to try to improve it. Matheuristics using ALNS were already used in other applications [Keskin and Çatay B., 2018] and showed the effectiveness of ALNS to diversify the search space. Another proposal of this dissertation is the addition of HBRKGA which is a BRKGA version, presented in Gonçalves and Resende [2011] in conjunction with the Local Branching. We also propose a reduction in the space as a maximum and minimum of facilities to be opened or closed. The components are assembled and tested in comparison to the literature methods.

The remainder of this dissertation is organized as follows: a detailed description and a mathematical formulation of TSCFL is presented in Chapter II, followed by the description of the proposed method in Section III. Computational experiments are described in Chapter IV in conjunction with comparisons between methods. Finally the conclusions are discussed and proposals for future works are presented in Chapter V.

Chapter II Mathematical definition and formulation for the TSCFLP

In this chapter we give the mathematical definition of the problem, define the formulation as presented in Litvinchev and Ozuna [2012] and present the methodology used in Louzada et al. [2016] to construct the approach for reducing the computational time of network update to calculate the minimum cost flow.

II.1 Problem Description and Mathematical Formulation

Given a set of factories and warehouses to be opened, the flows of products from factories to warehouses (level 1) and from warehouses to customers (level 2) are a typical scenario modeled by the Two-Stage Capacitated Facility Location (TSCFL). Also the TSCFL aims to make this logistic plan, while minimizing the total cost. Having said that, the TSCFL comes to facilitate the ambitions of reducing costs, saving resources and increasing profits, which is the objective of companies in general.

The main objective of the problem is to minimize costs, as long as the capacity constraints of factories and warehouses are met, and customers' demands are also satisfied. The costs mentioned above can be separated into opening costs and transportation costs. The opening costs are the cost to open (build/install) a factory or warehouse, while the transportation cost consists of the cost to transport products from factories to warehouses, and from warehouses to customers.

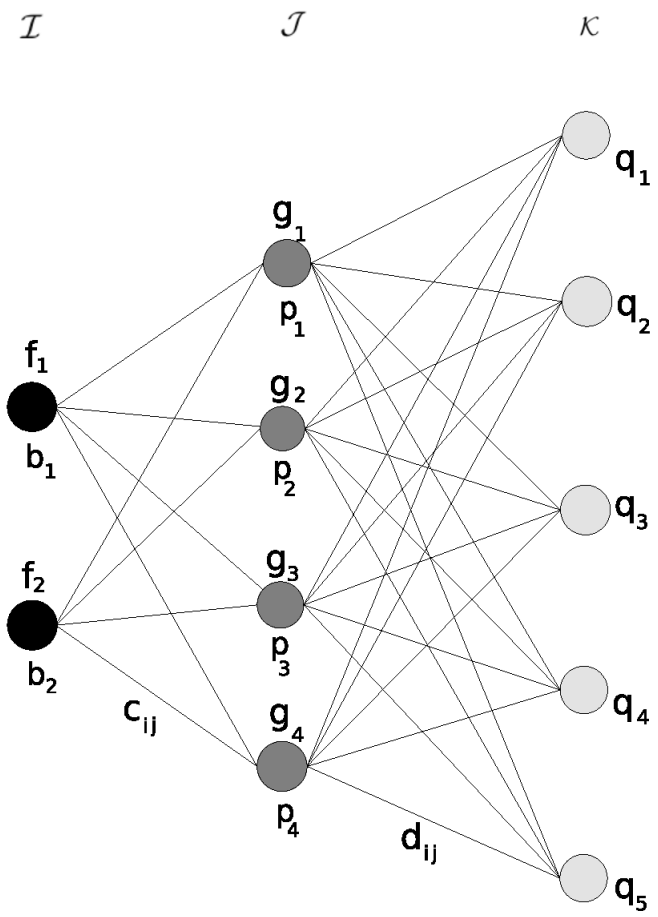

Figure II.1: Network with 2 factories, 4 warehouses and 5 customers.

As can be seen in Figure II.1, $\mathcal{I}$ represents the factories, $\mathcal{J}$ the warehouses, and $\mathcal{K}$ the set of all customers. For each factory $i \in \mathcal{I}$, its capacity b_i, its opening cost f_i and the costs c_{ij} of shipping products to all warehouses $j \in \mathcal{J}$ are defined. Similarly, for each warehouse $j \in \mathcal{J}$, its capacity p_j, its opening cost g_j and the costs d_{jk} of sending products to all customers $k \in \mathcal{K}$ are defined. For each customer $k \in \mathcal{K}$, q_k is defined as the demand to be satisfied.

Considering the definitions above, the TSCFL can be defined as determining a subset of factories $\bar{\mathcal{I}} \subseteq \mathcal{I}$ and warehouses $\bar{\mathcal{J}} \subseteq \mathcal{J}$ so that customer demands are met and the sum of opening and transportation costs is minimal.

In order to represent the TSCFL as Mixed Integer Programming Mixed Integer Programming (MIP) problem the decisions to be made at each step have to be defined in term of decision variables. Given that, let us define y_i, $i \in \mathcal{I}$ as decision variables that indicate whether the factory i will be opened or not. Also, let z_j, $j \in \mathcal{J}$ be defined as decision variables that indicate whether the warehouse j will be opened or not. At last, the decision variables x_{ij}, $i \in \mathcal{I}, j \in \mathcal{J}$, and s_{jk}, $j \in \mathcal{J}, k \in \mathcal{K}$, indicate how much flow is being sent from the factory i to the warehouse j and from the warehouse j to the customer k, respectively. Now, the TSCFL can be defined as a MIP problem as [Fernandes et al., 2014]:

$$\min \quad \sum_{i\in\mathcal{I}} f_i y_i + \sum_{j\in\mathcal{J}} g_j z_j + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{J}} c_{ij} x_{ij} + \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}} d_{jk} s_{jk}$$

$$\text{s.t:} \quad \sum_{j\in\mathcal{J}} s_{jk} = q_k \qquad \forall k \in \mathcal{K} \qquad \text{(II.1)}$$

$$\sum_{i\in\mathcal{I}} x_{ij} = \sum_{k\in\mathcal{K}} s_{jk} \qquad \forall j \in \mathcal{J} \qquad \text{(II.2)}$$

$$\sum_{j\in\mathcal{J}} x_{ij} \leq b_i y_i \qquad \forall i \in \mathcal{I} \qquad \text{(II.3)}$$

$$\sum_{k\in\mathcal{K}} s_{jk} \leq p_j z_j \qquad \forall j \in \mathcal{J} \qquad \text{(II.4)}$$

$$x_{ij} \in \mathbb{R}^+ \qquad \forall i \in \mathcal{I}, j \in \mathcal{J} \qquad \text{(II.5)}$$

$$s_{jk} \in \mathbb{R}^+ \qquad \forall j \in \mathcal{J}, k \in \mathcal{K} \qquad \text{(II.6)}$$

$$y_i \in \{0,1\} \qquad \forall i \in \mathcal{I} \qquad \text{(II.7)}$$

$$z_j \in \{0,1\} \qquad \forall j \in \mathcal{J} \qquad \text{(II.8)}$$

Constraints (II.1) ensure that each customer has their demand satisfied. Constraints (II.2) guarantee that the sum of the quantity of products that arrive in a warehouse needs to be equal to the quantity that this warehouse sends to customers. Also, Constraints (II.3) and (II.4) relate the flow sent by a factory (or warehouse) and whether they are open or not. Finally, Constraints (II.5) - (II.8) define the domain of each of the decision variables.

II.2 Network simplex and the minimum cost flow problem

As said in Section II.1, we have a flow from the factory to the warehouse (level 1) and another one from the warehouse to the client (level 2). For the methodology used in this dissertation proposal, we choose to use the network simplex to calculate which is an exact method for the minimum cost flow in the network. This minimal cost flow problem is necessary to calculate the total cost of a solution (presented in the objective function by the model in Section II.1)

To create the network is necessary to fill the capacity and cost of all the arcs of the facilities and to change this it is necessary to reconstruct the network. As shown in Figures II.2, II.3 to remove one factory from the solution it is necessary to remove one node and three edges to create a new network.

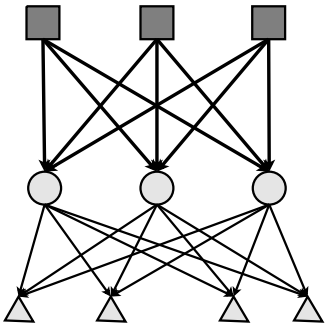

Figure II.2: Full Network 3 factories, 3 warehouses and 4 customers.

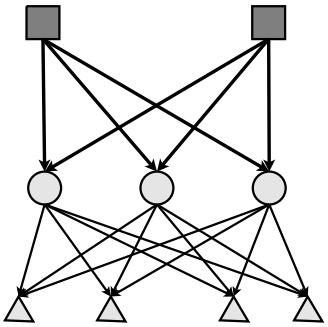

Figure II.3: Full Network 2 factories, 3 warehouses and 4 customers.

Since creating the network is an expensive computational process, we use the same approach presented in Louzada et al. [2016]. The network is constructed with one virtual node that has in its arcs the capacity of each factory and the same is made for the warehouses, one virtual node for each warehouse. The remainder of the network has fixed costs of transportation between factory and warehouse (level 1) and warehouse and clients (level 2). Then to update this network it is only necessary to change the capacity in the arcs of the virtual nodes when a capacity in the arc between the virtual node and the factory is zero, meaning it is closed. It is possible to update the arcs capacity with only $O(n)$ operations where n is the number of facilities, contrary to the other approach that was necessary $O(nm)$ operations, where n is defined as facilities and m is the number of connected facilities. To visualize this in a better way, Figures II.4 and II.5 replicate the examples from the last paragraph. The dashed line to the second factory represents the zero capacity arc updated.

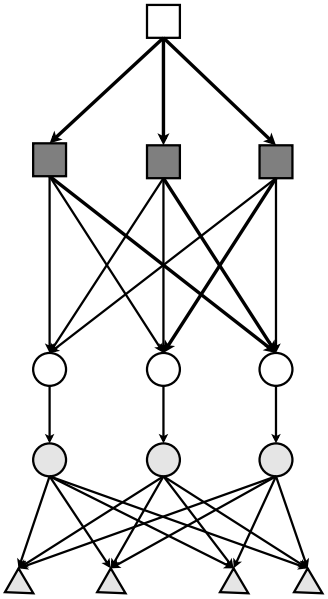

Figure II.4: Full Network 3 factories, 3 warehouses and 4 customers.

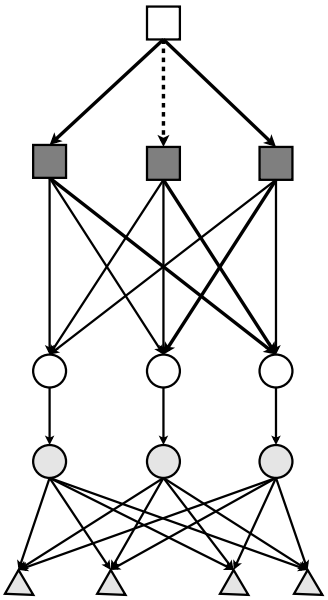

Figure II.5: Full Network 2 factories, 3 warehouses and 4 customers.

This flow problem is a linear programming problem, therefore it can be calculated with an exact method in polynomial time as shown in Orlin et al. [2001]. The structures presented are used as the basis for constructing the algorithms which will be presented in Chapter III.

Chapter III Algorithms for the TSCFLP

This section presents the main contribution of this dissertation proposal, which is a hybridization of Clustering Search - CS [Oliveira et al., 2013], Adaptive Large Neighborhood Search - ALNS [Ropke and Pisinger, 2006] and Local Branch - LB [Fischetti and Lodi, 2003], here called CS-ALNS-LB. The remainder of this section is organized as follows: constructive, destructive and repairing heuristic components are presented in Subsection III.1, followed by a local search procedure described in Subsection III.2. In Subsection III.3, the ALNS metaheuristic is presented. After that, in Subsection III.8, a Local Branching for the TSCFL is presented together with Section III.9 that presents a reduction space strategy for the Local Branching. Subsection III.10, the hybrid CS-ALNS-LB matheuristic is presented, uniting all components described in Chapter III. Then for last, in Sections III.11, III.12 and III.13, another metaheuristic is proposed as a version of Biased Random-key Genetic Algorithm (BRKGA) hybrid with Local Branch to create the architecture of Hybrid BRKGA(HBRKGA)

It is important to remark that a part of every metaheuristic is to calculate the value of a given solution's objective function. Although the opening costs are trivial to calculate, to calculate the transportation costs, a Minimum Cost Flow problem [Ahuja et al., 1988] has to be solved. From this point on, whenever necessary to calculate the transportation cost, except on Local Branching, we must consider the use of the *Network Simplex* algorithm of the CPLEX commercial package.

III.1 Semi-greedy Constructive Heuristic

The constructive heuristic proposed here to obtain a feasible solution for the TSCFL consists of: choosing a subset of factories to be opened; choosing a subset of warehouses to be opened; calculating the minimum cost flow.

The selection of a subset of facilities to be opened is carried out as follows: there is a repetition loop that chooses a factory randomly from a restricted candidates list, here called CL_1, until the sum of the capacities of the factories is greater than or equal to the sum of customers' demands. To build CL_1, the following equation was used as a cost-benefit ratio:

$$h_i^1 = \frac{f_i + \sum_{j \in \mathcal{J}} c_{ij}}{b_i}, \forall i \in \mathcal{I} \tag{III.1}$$

Once the cost-benefit ratio is defined, it is possible to build CL_1, according to the following rule:

$$CL_1 = \{i \in \mathcal{I} \setminus \mathcal{F} | h^1_{min} \leq h^1_i \leq h^1_{max} + \alpha(h^1_{min} - h^1_{max})\}, \tag{III.2}$$

where $h^1_{min} = \arg\min_{i \in \mathcal{I}}\{h^1_i\}$, $h^1_{max} = \arg\max_{i \in \mathcal{I}}\{h^1_i\}$, $\alpha \in [0, 1]$ and $\mathcal{F}$ it is the set of factories already open.

Similarly, for Step 2, a repetition loop is created in which a warehouse is chosen at each iteration until the sum of warehouse capacities is greater than or equal to the sum of customers' demands. Here, a restricted candidate list CL_2 is used considering the following cost-benefit ratio:

$$h^2_j = \frac{g_j + \sum_{k \in \mathcal{K}} d_{jk}}{p_j}, \forall j \in \mathcal{J} \tag{III.3}$$

Using the cost-benefit ratio above, CL_2 is formed using the following rule:

$$CL_2 = \{j \in \mathcal{J} \setminus \mathcal{D} | h^2_{min} \leq h^2_j \leq h^2_{max} + \alpha(h^2_{min} - h^2_{max})\}, \tag{III.4}$$

where $h^2_{min} = \arg\min_{j \in \mathcal{J}}\{h^2_j\}$, $h^2_{max} = \arg\max_{j \in \mathcal{J}}\{h^2_j\}$, $\alpha \in [0, 1]$ and $\mathcal{D}$ is the set of warehouses already open. Parameter α in Equations (III.2) and (III.4) defines how greedy or random the constructive approach, when near 0 less greediness otherwise more greediness.

In the last step, given the open facilities, the flow is obtained and the transportation and the opening costs are calculated, which together will be the value of the objective function of the problem. This sequence of steps is presented in detail in Algorithm 1:

Algorithm 1 Semi-greedy constructive - SGC

Require: $\mathcal{I}$,$\mathcal{J}$,$\mathcal{K}$, α
  $\mathcal{F} \leftarrow \emptyset$
  while $\sum_{i \in \mathcal{F}} b_i < \sum_{k \in \mathcal{K}} q_k$ **do**
    $CL_1^{min} \leftarrow GenerateCandidateList(\mathcal{I}, \mathcal{F}, \alpha)$
    $\mathcal{F} \leftarrow \mathcal{F} \cup rand(CL_1^{min})$
  end while
  $\mathcal{D} \leftarrow \emptyset$
  while $\sum_{j \in \mathcal{D}} p_j < \sum_{k \in \mathcal{K}} q_k$ **do**
    $CL_2^{min} \leftarrow GenerateCandidateList(\mathcal{J}, \mathcal{D}, \alpha)$
    $\mathcal{D} \leftarrow \mathcal{D} \cup rand(CL_2^{min})$
  end while
  $s_{best} \leftarrow CalculateFlow(\mathcal{F}, \mathcal{D})$

The semi-greedy constructive heuristic feasible solution is used to initialize the CS-ALNS-LB

and the population from HBRKGA. This solution diversifies population in HBRKGA. The next III.2 presents one of the components responsible for intensifying solution quality.

III.2 Local Search

Local searches are defined as a process of finding local minimums in a given neighborhood, using as input a previously defined solution. For this problem, three neighborhoods were defined: exchanging an open facility for a closed facility (Exchange Neighborhood); closing opened facilities (Close Neighborhood); and opening closed facilities (Open Neighborhood). In all these neighborhoods, the local search procedure used a best-improvement strategy.

To better understand the proposed neighborhoods, consider that each solution can be represented by a union of two disjoint sets, one containing the opened facilities ($\mathcal{O}$) and the other the closed ones ($\mathcal{C}$). Figures III.1, III.2 and III.3 presents each neighborhood and its characteristics. Also, in Figures III.1, III.2 and III.3, the dashed lines represent closed facilities ($\mathcal{C}$), while the solid lines represent the opened ones ($\mathcal{O}$).

The Exchange Neighborhood, represented in Figure III.1, starts by evaluating the exchange between each element of $\mathcal{O}$ and each element of $\mathcal{C}$. After evaluating each possible exchange, the pair of facilities $(\mathfrak{f}_i, \bar{\mathfrak{f}}_i)$, such that $\mathfrak{f}_i \in \mathcal{O}$ and $\bar{\mathfrak{f}}_i \in \mathcal{C}$ which leads to the best improvement of the inputted solution is selected.

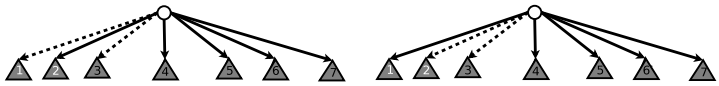

Figure III.1: Exchange - Open facility 1 and close facility 2.

The Close Neighborhood, represented in Figure III.2, consists in evaluate the removal of each element from $\mathcal{O}$ and adding it to $\mathcal{C}$. After evaluating the removal of each element from $\mathcal{O}$, the removal that leads to the best improvement of the inputted solution is selected.

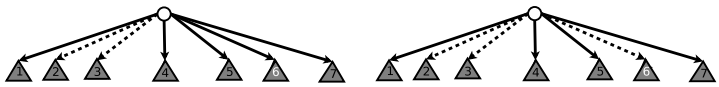

Figure III.2: Remove - Close facility 6.

The last neighborhood is the Open Neighborhood, represented in Figure III.3. It consists in evaluate the removal of each element from $\mathcal{C}$ and adding it to $\mathcal{O}$. After evaluating the addition of each element from $\mathcal{C}$ into $\mathcal{O}$, the addition, which leads to the best improvement of the inputted solution, is selected.

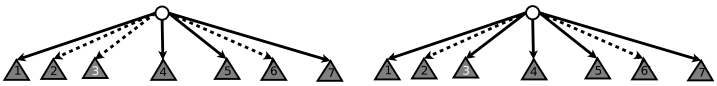

Figure III.3: Add - Open facility 3.

To reduce the number of explored solutions, a strategy defined in Louzada et al. [2016] was applied. This strategy consists of determining the maximum (β) and the minimum (π) number of factories and warehouses needed to guarantee that all customers' demands can be fulfilled. Having that, one may avoid exploring solutions with less than the minimum number of required facilities and solutions with more than the maximum number of required facilities. Avoid exploring these solutions is very important since evaluating a solution of the TSCFL is computationally expensive.

In order to determine β, for each type of facility, the following procedure was executed. First, the facilities are sorted in increasing order of capacities. After that, to define the maximum number needed of the chosen type of facility, a subset of facilities is created. This subset is created by adding facilities, from the facility with smaller capacity to the one with the maximum capacity, until the summation of capacities in this subset becomes greater or equal to the summation of customers' demands. Once this process is done, the obtained subset's cardinality is the maximum number needed for the chosen type of facility. The process to obtain π is similar to the process for determining β. The difference is that one needs to create the subset starting from the facility with maximum capacity and keep going in decreasing order until the summation of capacities in this subset becomes greater or equal to the summation of customers' demands. This strategy is efficient for both Close and Open neighborhoods, and so it was employed on both of them.

All the presented neighborhoods were organized in a Random Variable Neighborhood Search (RVND) framework [Penna et al., 2013] and it is described in Algorithm 2:

Algorithm 2 RVND

Require: Solution s
  $N \leftarrow$ Random(Neighborhood $N^{swap}, N^{close}, N^{open}$)
  $k \leftarrow 1$
  $s_{best} \leftarrow s$
  while $k \leq 3$ **do**
    Find best neighbor $s' \in N^k(s)$
    if $f(s') < f(s_{best})$ **then**
      $s_{best} \leftarrow s'$
      $k \leftarrow 1$
    else
      $k \leftarrow k + 1$
    end if
  end while
  return (s_{best})

The RVND framework was chosen because it is difficult to determine and define the best order

for exploring the neighborhoods. In Algorithm 2, N is defined as a vector of neighborhoods and N^k is the k-th position in this shuffled vector. The neighborhood order is not just problem related but also instance-related, so that a random order may be more effective than a fixed order. Given that, each time the RVND procedure starts, a new order is randomly generated. The RVND procedure is used to exploit the solution space in the CS-ALNS-LB and HBRKGA.

III.3 Adaptive Large Neighborhood Search - ALNS

The Adaptive Large Neighborhood Search (ALNS) was proposed by Ropke and Pisinger [2006] and combines a constructive heuristic with several destructive and repair procedures, in an adaptive way, to explore different regions of the solution space. An important feature of the ALNS is that the repair procedures, the destructive procedures, or both the repair and destructive procedures have their selection based on its success. Considering that the ALNS components have to be built specifically for each problem, the destructive and repair processes for the TSCFL are defined in the sections III.4,III.5 e III.6.

III.4 Random Destructive

The Random Destructive works as follows: Given a percentage for destruction, factories and warehouses are randomly closed. Regardless of the cost of the solution, facilities are closed without distinction. As a result, the Random Destructive is an important component to generate diversification.

III.5 Semi-Greedy Destructive

The Semi-Greedy Destructive uses multiple cost-benefit ratios and the RCL concept of the Semi-greedy Constructive Heuristic, presented in Section III.1. In addition to the cost-benefit ratio defined in Section III.1, two other cost-benefit ratios were proposed.

The first one takes into consideration the average transportation costs from factories to warehouses, Equation (III.5), and from warehouses to customers, Equation (III.6).

$$h_3 = \frac{f_i + \frac{1}{|\mathcal{J}|}\sum_{j\in\mathcal{J}} c_{ij}}{b_i}, \forall i \in \mathcal{I} \tag{III.5}$$

$$h_4 = \frac{g_j + \frac{1}{|\mathcal{K}|}\sum_{j\in\mathcal{J}} d_{jk}}{b_i}, \forall j \in \mathcal{J} \tag{III.6}$$

The second cost-benefit ratio proposed for the Semi-greedy Destructive procedure considers the opening cost of the facilities divided by their respective capacities. Equations (III.7) and (III.8)

formally describe this idea:

$$h_5 = \frac{f_i}{b_i}, \forall i \in \mathcal{I} \tag{III.7}$$

$$h_6 = \frac{g_j}{p_j}, \forall j \in \mathcal{J} \tag{III.8}$$

These three cost-benefit ratios were used in an attempt to seek diversity, since it is impossible to know whether or not one will be better than the other for each problem instance. This situation arises since different real scenarios lead to different instances, which makes it impossible to guarantee, a priori, if any cost (or which cost) dominates the others.

The Semi-greedy Destructive procedure randomly selects one of the cost-benefit ratios and creates a candidate list composed of the factories or warehouses identifiers, and its respective cost-benefit values. This list is restricted by Equation (III.9):

$$CL^{max} = \{i \in \mathcal{I} \setminus \mathcal{F} | h^1_{max} \geq h^1_i \geq h^1_{min} + \delta(h^1_{max} - h^1_{min})\} \tag{III.9}$$

This function is similar to that used to restrict candidates in the Semi-greedy Constructive procedure. However, instead of minimizing, now the facilities with maximum cost-benefit are desired. The idea is to choose candidates with high opening costs, but not necessarily the highest, to generate diversity.

Algorithm 3 presents the details of the Semi-greedy Destructive procedure. The parameters σ, δ and *random* are, respectively, the rate of destruction, the value that defines the size of the cost-benefit selection window, and whether the adaptive used is random or semi-greedy.

Algorithm 3 Semi-greedy Destrucitve - SGD

Require: $\mathcal{I}$,$\mathcal{J}$,$\mathcal{K}$, δ ,s_{temp},h_{random},σ, $random$

$s_{destroy} \leftarrow s_{temp}$
$max_{destroy} \leftarrow \sum_{i \in \mathcal{I}} s_{destroy} * \sigma$
$destroy \leftarrow 0$
while $destroy < max_{destroy}$ **do**
  $CL^{max}_{h_{random}} \leftarrow GenerateCandidateList(\mathcal{I}, \mathcal{F}, \delta, random)$
  $s_{destroy} \leftarrow s_{destroy} \setminus rand(CL^{max}_{h_{random}})$
  $destroy \leftarrow destroy + 1$
end while
$max_{destroy} \leftarrow \sum_{j \in \mathcal{J}} s_{destroy} * \sigma$
$destroy \leftarrow 0$
while $destroy < max_{destroy}$ **do**
  $CL^{max}_{h_{random}} \leftarrow GenerateCandidateList(\mathcal{J}, \mathcal{D}, \delta, random)$
  $s_{destroy} \leftarrow s_{destroy} \setminus rand(CL^{max}_{h_{random}})$
  $destroy \leftarrow destroy + 1$
end whilereturn $s_{destroy}$

When the algorithm receives the $random = 0$, the cost-benefit ratios are ignored, so facilities are randomly selected. Otherwise, when $random = 1$, the benefit-cost functions described above are used. Thus the *random* parameter is responsible for creating the dynamic between the random and semi-greedy. The Semi-greedy Destructive procedure may lead to infeasible solutions, which are repaired using the Semi-Greedy Repairer defined in Section III.6.

III.6 Semi-Greedy Repairer

The Semi-Greedy Repairer is a procedure that receives an infeasible solution and returns a feasible one. In order to fulfill its role, a cost-benefit ratio is selected at random from the set of cost-benefit ratios defined in Section III.5. Then, a Restricted Candidate List is generated using the same process of the Semi-greedy Constructive procedure. This process is presented in detail in Algorithm 4:

Algorithm 4 Semi-greedy Repairer - SGR

Require: $\mathcal{I}$,$\mathcal{J}$,$\mathcal{K}$, α_2 ,$s_{destroy}$,h_{random}

$s_{repair} \leftarrow s_{destroy}$

while $\sum_{i \in s_{repair}} b_i < \sum_{k \in \mathcal{K}} q_k$ **do**

$CL_{h_{random}} \leftarrow GenerateCandidateList(\mathcal{I}, \mathcal{F}, \alpha_2)$

$s_{repair} \leftarrow s_{repair} \cup rand(CL_{h_{random}})$

end while

while $\sum_{j \in s_{repair}} p_j < \sum_{k \in \mathcal{K}} q_k$ **do**

$CL_{h_{random}} \leftarrow GenerateCandidateList(\mathcal{J}, \mathcal{D}, \alpha_2)$

$s_{repair} \leftarrow s_{repair} \cup rand(CL_{h_{random}})$

end while

return s_{repair}

It is important to note that the indexes used to create the candidate lists are h_{random}, and this value is set before the Semi-greedy Repairer selects the cost-benefit ratio at random. Parameter α_2 defines how greedy or random the repairer approach is.

III.7 ALNS Structure for the TSCFL

The ALNS for the TSCFL combines the components described in the previous sections, using the following structure: First, a solution is built using Semi-greedy Constructive (SGC) and then an iterative process of destruction Semi-greedy Destructive (SGD) and repair Semi-greedy Repairer (SGR) is repeated max_{iter} times. In the first iteration, the probability of choosing each of destructive strategy is the same. Every time the repair procedure finds a new global best solution, the probability of selecting the destructive used before the repairing is incremented. This probability is used as a mechanism to prioritize strategies that are behaving better for the current instance.

This process is described in Algorithm 5:

Algorithm 5 Adaptive Large Neighborhood Search

Require: $\mathcal{I}, \mathcal{J}, \mathcal{K}, \alpha, \delta, \alpha 2, \sigma, s, s_{best}$,$probability$, max_iter
  $s \leftarrow SGC(\mathcal{I}, \mathcal{J}, \mathcal{K}, \alpha)$
  $s_{best} \leftarrow s$
  while $i < max\_iter$ **do**
    $random \leftarrow RandomProbability(0, 1, probability)$
    $h_{random} \leftarrow Random \in$ **card**(cost-benefit ratios)
    $s \leftarrow SGD(\mathcal{I}, \mathcal{J}, \mathcal{K}, \delta, s, h_{random}, \sigma, random)$
    $s \leftarrow RSG(\mathcal{I}, \mathcal{J}, \mathcal{K}, \alpha 2, s, h_{random}, random)$
    UpdateProbability$(s_{best}, s, probability, random)$
  end while
  return s_{best}

The two functions, *RandomProbability* and the *UpdateProbability* are responsible for the adaptive procedure, the first one compares the probability with a random number between zero and one, if it's less than the probability it returns the Random Destructive, otherwise the Semi-greedy Destructive is returned. If a best solution is founded, the *UpdateProbability* increases in 0.1 the probability of choosing such a method, and decreases in 0.1 the probability of choosing another method.

III.8 Local Branching

The Local Branching (LB) technique, introduced by Fischetti and Lodi [2003], can be used in a way that mixed-integer programming models can be used to improve a given feasible solution. That said, one may say that LB uses a mixed-integer linear programming solver to explore the solution space. This procedure can be correlated to a local search procedure, in which the neighborhoods are defined through the introduction of linear inequalities in the mixed-integer linear programming model. That strategy has already been used in the literature and provided good results [Gonzalez et al., 2016; Gonzalez and Brandão, 2018].

Specifically for the TSCFL, let us consider a solution $s = \langle \bar{y}, \bar{z} \rangle \in P$, where P is the polyhedron formed by the Constraints (II.1) - (II.8). The idea consists of adding the following LB constraints:

$$\sum_{i \in I | \bar{y}_i = 0} y_i + \sum_{i \in I | \bar{y}_i = 1} (1 - y_i) \leq \Delta_1, \tag{III.10}$$

$$\sum_{j \in J | \bar{z}_j = 0} z_j + \sum_{j \in J | \bar{z}_j = 1} (1 - z_j) \leq \Delta_2, \tag{III.11}$$

where Δ_1 and Δ_2 are a given positive integer, indicating the number of variables y_i, $i \in \mathcal{I}$ and z_j, $j \in \mathcal{J}$, which can have their value changed from one to zero and vice versa.

III.9 Changes in the Local Branch procedure

As we use limits to the number of opened facilities, for the local search procedure explained in Section III.2, we want to conduct an investigation to understand the effect of reducing search space during the Local Branch procedure. It is important to remember that this procedure can only be applied because it is an heuristic method, meaning that the task is not to certify optimality.

This application consists of using β_i (maximum number of facilities) and π_i (minimum number of facilities), explained in Section III.2, as part of the Local Branch where $i \in \{1, 2\}$, factories and warehouses respectively. This addition is formalized as Equations III.12 - III.15.

$$\sum_{i \in I} y_i \leq \beta_1 \tag{III.12}$$

$$\sum_{i \in I} y_i \geq \pi_1 \tag{III.13}$$

$$\sum_{j \in J} z_j \leq \beta_2 \tag{III.14}$$

$$\sum_{j \in J} z_j \geq \pi_2 \tag{III.15}$$

We have to apply these four equations as part of the Local Branch procedure and test them for solution quality and computational time. This application has to be tested for the HBRKGA and CS-ALNS-LB.

III.10 CS-ALNS-LB Matheuristic

The Clustering Search metaheuristic (CS) was proposed by Oliveira and Lorena [2007] as a generic way of combining metaheuristics and clustering to identify promising regions of the space to apply local search procedures. As calculating the cost function for this problem is computationally expensive, the CS hybrid metaheuristic is suited for this problem, because it only applies the local search procedure when several solutions in the same promising region are obtained.

As mentioned before in the literature [Oliveira and Lorena, 2007; Chaves et al., 2016; Rosa et al., 2016; González et al., 2019a], the CS metaheuristic needs another technique to generate feasible solutions scattered in different regions of the solution. Any metaheuristic can be employed to generate solutions into CS. Oliveira et al. [2013] reports several successful approaches including Simulated Annealing (SA), Genetic Algorithm (GA) and GRASP. However, to the best of our knowledge, ALNS has not yet been used with CS for any problem. In addition, ALNS has shown

good results for different problems outperforming other traditional metaheuristics such as SA and GA [Laporte et al., 2010; Ribeiro et al., 2012; Mauri et al., 2016; Kiefer et al., 2017].

Having stated that, in this dissertation proposal the ALNS presented in Section III.7 was chosen to generate solutions into CS. The CS-ALNS works by, initially, separating each obtained solution in different clusters, until γ clusters are defined. After that, for each solution found we check in which cluster it should be placed, by using the Hamming distance to calculate the distance between the new solution and the cluster' *center*, to decide which cluster.

Once the cluster is decided, if the solution has a better cost than the center, it becomes the new center of the cluster, and the volume of the cluster is updated. Otherwise, just the volume of the cluster is updated. When that volume reaches a maximum value, max_{volume}, the local search process (described in Section III.2) starts using the cluster center as incumbent solution. Another mechanism for escaping from local optimum is to verify whether a cluster fails to improve the best solution found after max_{inef} applications of the local search. Whenever that happens, the center of the cluster goes through a process of destruction and semi-greedy repair.

To complete the Matheuristics approach a mathematical modeling method, the Local Branching (described in Section III.8), in conjunction with heuristics Clustering Search and ALNS to the best solution obtained by the CS-ALNS. The whole procedure is presented in Algorithm 6:

Algorithm 6 CS-ALNS-LB

Require: $\mathcal{I}, \mathcal{J}, \mathcal{K}, \alpha, \alpha_2, \delta, \gamma, max_{volume}, max_{inef}, \sigma$
  $C_i \leftarrow 0, v_i \leftarrow 0, r_i \leftarrow 0 | i \in \{1..\gamma\}$
  $clt \leftarrow 0$
  $s_{best} \leftarrow SGC(\mathcal{I}, \mathcal{J}, \mathcal{K}, \alpha)$
  $s \leftarrow s_{best}$
  $probability \leftarrow \frac{1}{2}$
  while $i < max\_iter$ **do**
    ALNS($\mathcal{I}, \mathcal{J}, \mathcal{K}, \alpha, \delta, \alpha 2, \sigma, s, s_{best}$,$probability$, 1)
    if $clt < \gamma$ **then**
      $C_i \leftarrow s$,$v_i \leftarrow 1$, $r_i \leftarrow 0$, $clt \leftarrow clt + 1$
    else
      $i \leftarrow \arg\min_{i \in \{1..\gamma\}} H_i$, $v_i \leftarrow v_i + 1$
      UpdateCenter(C_i,s)
      if $v_i = max_{volume}$ **then**
        $s \leftarrow RVND(C_i)$
        $s \leftarrow Update(s_{best}, C_i)$
        if C_i was modified **then**
          $r_i \leftarrow 0$
        else
          $r_i \leftarrow r_i + 1$
        end if
        if $r_i = r_{max}$ **then**
          $C_i \leftarrow SGD(\mathcal{I}, \mathcal{J}, \mathcal{K}, \delta, C_i, h_{random}, \sigma, random)$
          $C_i \leftarrow SGR(\mathcal{I}, \mathcal{J}, \mathcal{K}, \delta, C_i, h_{random}, \sigma, random)$
        end if
      end if
    end if
  end while
  $s_{best} \leftarrow$ LocalBranching(s_{best})
  return s_{best}

The function UpdateCenter receives the new solution that is going to enter the cluster and the solution defined as the center of the cluster. Whenever this new solution has a better cost than the solution defined as cluster center, the new solution becomes the center of the cluster and the old center is discarded.

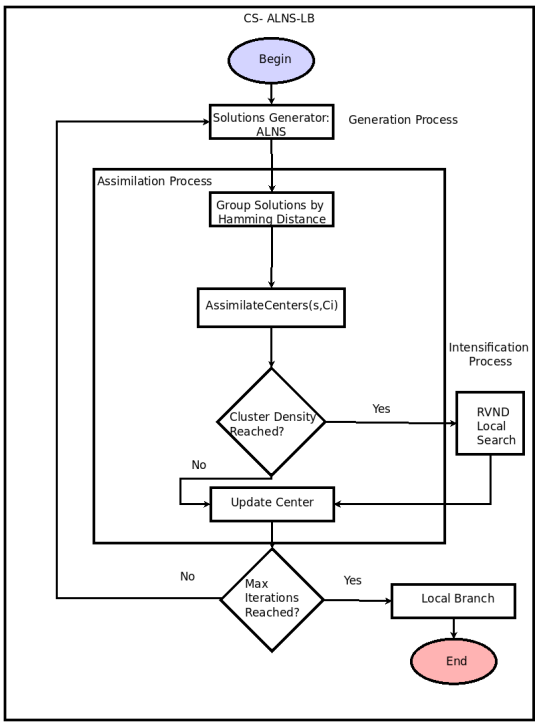

Figure III.4: CS-ALNS-LB flow diagram. Adaptation from Oliveira et al. [2013]

Another meta-heuristic is proposed as a test to compare with the CS-ALNS-LB with and without the reduction in space explained in Section III.11.

III.11 Biased Random-Key Genetic Algorithm

The Random-key Genetic Algorithm Random-key Genetic Algorithm (RKGA) was first elucidated by Bean [1994] for combinatorial optimization problems whose solutions were easily represented by permutations of vectors. The work's main contribution was related to the concept of random keys, where genes in a chromosome are encoded to random values in [0,1] interval that generates a similar effect to a random search in conjunction with the Genetic Algorithm. This idea has been used in many sequencing problems.

The heuristic implemented as a base for the proposed method to solve the TSCFL was the BRKGA, Biased Random Key Genetic Algorithm (Gonçalves and Resende [2011]), which is a variation of RKGA that has a difference in the way parents are selected for the crossover. In BRKGA, chromosomes (solutions) are represented by real values obtained from range [0,1], in the same manner of RKGA. A decoder is then applied to each of these chromosomes to transform the random-key vector into a solution of the combinatorial optimization problem and calculate the value of the objective function. It is worth mentioning that the decoder is specific to each optimization problem.

After the decoder stage, the solutions are separated into two sets, elite and non-elite, and

BRKGA uses an elitism strategy to generate a new population. Each new element of the population is generated either by crossover or mutation. When applying the crossover, new solutions are obtained by combining two solutions. The crossover is called biased because one parent is always an elite individual and has a higher probability of passing its genes (characteristic) to the next generation. The mutation process consists of randomly changing random keys from elements of the population to generate diversity.

With this in mind we propose to the population the use of the same constructive used for the CS-ALNS-LB and in the next section we explain our architecture for the decoding process that will be used for the TSCFL.

III.12 Decoder

The decoder is a deterministic algorithm whose purpose is to receive a given chromosome of random-keys as input and associate it with a solution of the combinatorial optimization problem, for which an objective value or fitness will be computed.

As we already know, the decoder receives a chromosome, i.e., a vector of random-keys. The decoding process for this problem is composed of five steps, as see in Fig. III.5

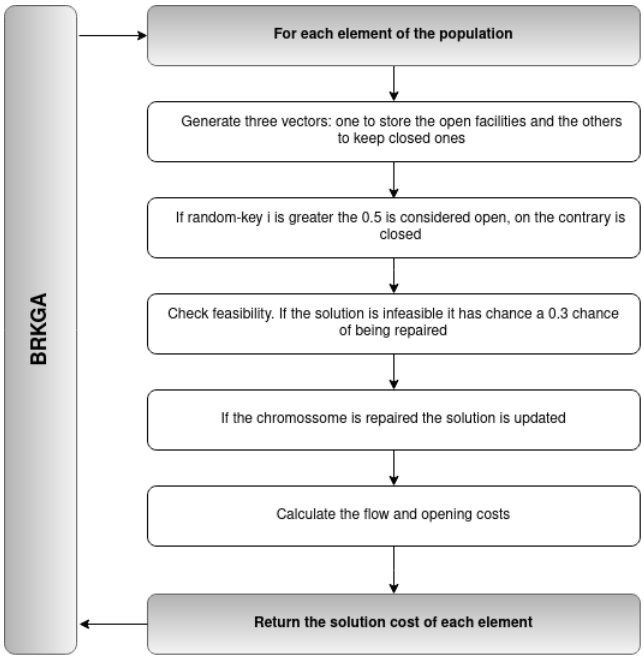

Figure III.5: Decoder for TSCFL problem

A representation of a TSCFL solution can be defined as a binary vector, with a size equal to the number of factories and warehouses, where zero represents that a facility is closed and one represents that a facility is considered open.

For each element of the population, the Decoder receives its chromosome and generates three vectors. The first one to store open facilities, while the others are to keep the identifier of closed

factories and warehouses.

Then, for each element of the chromosome, if the element is greater than 0.5, the facility is considered open and its capacity is stored. Otherwise, it is considered closed and its identifier is stored in the correspondent closed facilities vector.

In the third step, if the solution represented by the received chromosome is infeasible, the solution has a 0.3 probability of being repaired. This process uses the closed facilities vector and randomly selects some until the demand is satisfied. The choice of the 0.3 value was made to control diversity in the chromosomes, reducing the number of infeasible solutions.

If the infeasible solution was repaired, its chromosome is updated to comprise the new open facilities during the fourth step.

In the last step, the Decoder calculates the flow of products through the opened factories and warehouses, returning the solution cost for the BRKGA method.

With the defined decoder we can construct the HBRKGA approach to the TSCFL. It will be explained in details in the next section.

III.13 Architecture of the HBRKGA to the TSCFL

The Hybrid BRKGA (HBRKGA) consists of a straightforward combination of the previously presented components (SGC, BRKGA, RVND, and Local Branching). The BRKGA's (described in Sections III.11 and III.12) population is generated using SGC (Section III.1), and after its series of generations, two intensification steps are executed. Using the BRKGA best-known solution, the RVND explores all the neighborhoods described in Section III.2. Using the local optimum solution obtained by the RVND as a base solution, a Local Branching is executed following the description presented in Section III.8 in conjunction with SectionIII.9. HBRKGA's workflow can be seen in Fig. III.6.

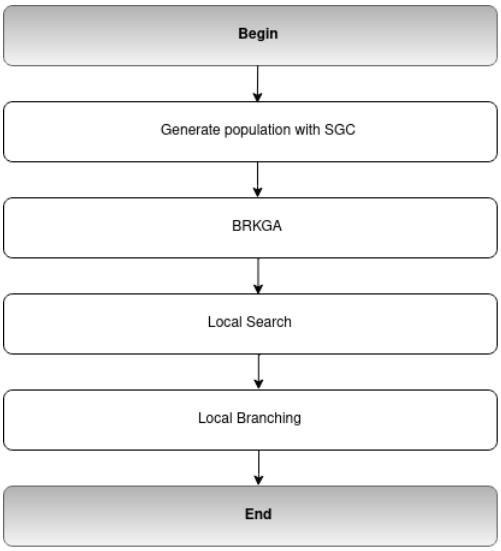

Figure III.6: HBRKGA Workflow

All the tests of this two methodologies are presented in the remainder of this dissertation. The reduction in space proposed is tested for both methods.

Chapter IV Computational results and conclusions for the TSCFLP

In this chapter we present the computational results and make a comparison between the methods presented in the literature for the same problem.

The proposed methods (CS-ALNS-LB and HBRKGA) were programmed in C++, using a GCC compiler version 7.5.0, and were executed ten times for each instance on an Intel Core i5 processor with a frequency of 2.30 GHz and 16 GB RAM.

The instances used were defined in Fernandes et al. [2014] and are chosen for the purpose of comparison with the literature, they can be divided into two sets. The first set consists of 25 instances with 50 factories, 100 warehouses and 200 customers. In the second set, also with 25 instances, the size of the problem is doubled, which means 100 factories, 200 warehouses and 400 customers. Both sets are divided into five instance classes with different characteristics.

All instances can be find at https://github.com/pehgonzalez/OCA, along with the binary file to reproduce the experiments. In order to do a fair comparison regarding the computational time between the proposed approach and the methods reported in the literature, a conversion factor was applied. Thus, the Single Thread Rating reported for each CPU, available at https://www.cpubenchmark.net/, was used. The corresponding factors are presented in Table IV.1.

GA[a]	HEA/FA[b]	BRKGA+LS[c]	GRASPH[d]
0.46	1.04	0.83	1.18

Table IV.1: CPU conversion factor.
[a] Fernandes et al. [2014] - Pentium Intel 2,3GHz, 24GB de RAM
[b] Guo et al. [2017] - Intel Core i7 3.6GHz, 8GB de RAM
[c] Biajoli et al. [2019] - Intel Core i5 1.7GHz, 16GB de RAM
[d] Gonzalez et al. [2019b] - Intel Core i7 4.2GHz, 12GB de RAM

In this section we will first evaluate the CS-ALNS-LB and then we will evaluate the HBRKGA against other methods, for last we will make a comparison between the HBRKGA and CS-ALNS-LB against the most successfull methods in the literature.

The remainder of this section is organized into two subsections. The first one presents the computational results and discuss them. After that, a statistical analysis is done to verify whether the proposed approach is significantly different from the others in the literature.

IV.1 Computational Results - CS-ALNS-LB

To determine the values for each parameter of the CS-ALNS-LB , the iRace [Manuel et al., 2016] package was used. As a fixed parameter for the LB, 50 seconds was used as a stopping criterion for the CPLEX solver. Table IV.2 presents the values tested for each parameter of the CS-ALNS-LB and their values used in the experiments.

Table IV.2: Values tested and selected for each parameter of the CS-ALNS-LB.

Parameters	Intervals	Selected Values
α	$\{0.60; 0.65; 0.70; 0.75; 0.80; 0.85; 0.90;\}$	0.75
δ	$\{0.60; 0.65; 0.70; 0.75; 0.80; 0.85; 0.90;\}$	0.90
α_2	$\{0.60; 0.65; 0.70; 0.75; 0.80; 0.85; 0.90;\}$	0.90
γ	$(5, 20)$	19
max_{inef}	$(5, 20)$	18
max_{volume}	$(5, 15)$	10
max_{iter}	$[80, 400]$	260
σ	$\{0.1; 0.20; 0.30; 0.40\}$	0.40

Tables IV.3 and IV.4 present the results obtained by CS-ALNS-LB, along with the results obtained by the other methods in the literature for the TSCFL. In both tables, the first columns indicate the class of the instance, the identifier and the best known lower bound (LowerB), presented by Guo et al. [2017]. As for the CS+CPLEX method, the authors of Louzada et al. [2016] provided us with the code, which allowed us to rerun it in the same computational environment where the CS-ALNS-LB experiments were performed.

Columns *AVG* and *BST* are calculated as distance from the Lower Bound presented by Guo et al. [2017], measured as $\frac{sol-LowerB}{LowerB} \times 100$, where *sol* indicates the *AVG* or *BST* solutions from each method. The column *Time* is calculated as the mean time in seconds for 10 executions. The method of the CS-ALNS-LB is presented in two forms (woRD) without Reduction of the Space and the other is with it.

Table IV.3: Results obtained for the first set of instances (50 factories, 100 warehouses and 200 customers).

Instances			GA		CS+CPLEX			HEA/FAc			BRKGA+LS			GRASPH			CS-ALNS-LB woRD			CS-ALNS-LB wRD		
Class	Id	LB	AVG	Time	Best	AVG	Time	Best	AVG	Time	Best	AVG	Time	Best	AVG	Time	Best	AVG	Time	Best	AVG	Time
	1	721209.6	**0.13**	264.78	**0.13**	0.22	48.90	**0.13**	0.24	431.34	0,14	0.14	19.32	**0.13**	**0.13**	**4.50**	**0.13**	**0.13**	15.41	**0.13**	**0.13**	17.56
	2	730451.6	0.40	257.17	**0.23**	0.31	76.86	0.31	0.36	510.74	0,24	0.24	115.93	0.24	0.24	21.34	**0.23**	**0.23**	**21.02**	**0.23**	**0.23**	28.75
1	3	731885.3	0.24	263.35	**0.21**	0.29	51.94	0.22	0.24	327.60	0,22	0.22	39.29	0.22	0.22	**30.46**	**0.21**	**0.21**	55.22	**0.21**	**0.21**	69.73
	4	721515	0.81	242.93	1.19	1.41	96.62	0.54	0.59	472.15	0,51	0.51	116.29	**0.50**	**0.50**	29.08	**0.50**	**0.50**	**18.98**	**0.50**	**0.50**	20.57
	5	713633.8	0.82	251.79	**0.81**	0.88	**56.14**	0.86	0.97	480.59	**0,81**	**0.81**	59.10	**0.81**	**0.81**	160.01	**0.81**	**0.82**	64.69	**0.81**	**0.82**	67.17
	1	479860.2	2.69	144.39	**2.68**	3.27	27.69	2.74	2.82	239.25	2,69	2.69	16.84	2.69	2.69	383.27	**2.68**	**2.68**	**15.43**	**2.68**	**2.68**	17.12
	2	483072.2	**2.30**	144.16	**2.30**	2.62	81.36	2.34	2.41	206.37	2,31	2.31	**60.58**	**2.30**	2.34	368.58	**2.30**	**2.30**	64.89	**2.30**	**2.30**	76.77
2	3	486018.5	2.14	150.60	**1.86**	2.03	**30.82**	1.87	1.89	173.38	1,87	1.87	117.20	1.88	1.94	590.94	**1.86**	**1.86**	48.02	**1.86**	**1.87**	72.38
	4	482374.6	2.04	142.25	**2.01**	**2.01**	83.02	2.07	2.12	168.04	2,05	2.05	59.59	2.02	2.02	365.57	**2.01**	**2.02**	**47.37**	**2.01**	**2.02**	53.20
	5	474803.3	3.14	126.08	**3.12**	3.39	36.98	**3.12**	**3.12**	176.79	**3,12**	**3.12**	**10.84**	**3.12**	**3.12**	590.87	**3.12**	**3.12**	33.33	**3.12**	**3.12**	25.38
	1	2608800	**3.07**	125.90	**3.07**	**3.07**	**19.89**	3.14	3.30	121.36	3,11	3.11	126.94	3.22	3.30	596.03	**3.07**	**3.10**	104.13	**3.07**	**3.13**	94.08
	2	2616252	3.12	130.22	**3.10**	**3.10**	73.09	3.30	3.36	120.92	3,17	3.17	**40.26**	3.37	3.39	594.73	3.13	**3.20**	94.18	**3.13**	**3.16**	89.40
3	3	2598277	3.11	123.56	**3.09**	**3.10**	**52.98**	3.30	3.39	167.70	**3,1**	**3.10**	72.94	3.23	3.32	591.33	**3.09**	**3.14**	77.21	**3.09**	**3.16**	93.12
	4	2612534	3.07	107.73	**3.05**	**3.05**	45.21	3.18	3.22	154.29	3,1	3.10	**8.25**	3.18	3.29	593.68	**3.05**	**3.10**	80.60	**3.05**	**3.11**	91.74
	5	2568856	**3.01**	110.36	**3.01**	**3.01**	47.45	3.17	3.19	163.91	3,13	3.13	**40.23**	3.14	3.22	593.27	**3.01**	**3.03**	76.79	**3.01**	**3.02**	91.92
	1	525294.1	**3.14**	138.25	**3.14**	3.60	89.47	3.36	3.54	224.71	3,22	3.22	58.44	3.29	3.29	591.54	**3.14**	**3.14**	**39.42**	**3.14**	**3.14**	51.83
	2	526911.7	**2.33**	139.83	**2.43**	2.71	102.38	2.74	2.92	206.66	2,51	2.51	80.98	2.65	2.80	592.19	**2.43**	**2.45**	**57.23**	**2.43**	**2.45**	71.77
4	3	532592.3	2.66	144.88	2.41	2.44	118.38	2.59	2.62	256.30	**2,42**	**2.42**	110.29	2.45	2.92	591.35	**2.30**	**2.44**	**67.49**	**2.38**	**2.48**	90.61
	4	529372	2.53	127.30	**2.35**	2.66	133.69	2.63	2.81	271.29	2,51	2.51	74.85	2.36	2.50	591.31	**2.35**	**2.36**	**55.50**	**2.35**	**2.40**	79.13
	5	521470.1	3.13	120.27	3.15	3.53	115.67	3.18	3.18	188.88	3,16	3.16	**39.57**	3.15	3.23	388.72	**3.12**	**3.12**	46.46	**3.12**	**3.13**	78.55
	1	2743547	1.20	164.42	1.19	1.19	157.20	**1.16**	1.20	206.49	1,22	1.22	120.41	1.24	1.31	591.33	**1.16**	**1.18**	**64.60**	**1.16**	**1.22**	75.05
	2	2752021	**1.07**	156.71	1.08	1.11	89.13	1.11	1.16	269.19	1,09	1.09	**35.71**	1.15	1.17	591.31	**1.07**	**1.07**	68.15	**1.07**	**1.07**	71.02
5	3	2737769	**1.10**	191.60	**1.09**	**1.10**	126.33	1.22	1.28	210.10	1,15	1.15	130.43	1.29	1.30	591.78	**1.09**	**1.13**	**70.37**	**1.09**	**1.15**	70.92
	4	2748216	**1.07**	136.87	**1.05**	1.12	149.59	1.10	1.13	210.73	1,1	1.10	151.61	1.06	**1.07**	591.67	**1.05**	**1.07**	**67.09**	**1.05**	**1.07**	68.39
	5	2702350	1.25	145.07	1.24	**1.24**	**54.96**	1.31	1.34	188.84	1,27	1.27	74.20	1.29	1.34	592.50	1.23	**1.25**	67.90	**1.22**	**1.24**	72.33
Average			1.98	162.02	1.96	2.10	78.63	2.03	2.10	245.91	1.95	1.97	71.20	2.00	2.06	449.09	**1.93**	**1.95**	**56.86**	**1.93**	**1.95**	65.54

[a] Fernandes et al. [2014] [b] Guo et al. [2017]
[c] Biajoli et al. [2019] AVG, BST in percentage and Time in seconds
[d] Gonzalez et al. [2019b]

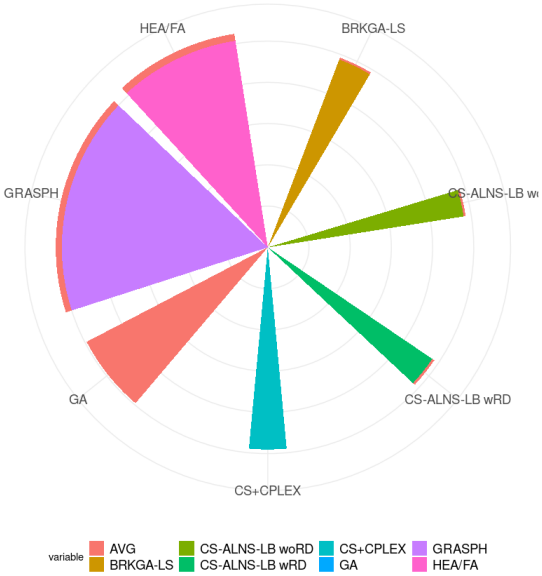

Figure IV.1: Summary of the Results for Instances of 50 Factories.

Figures IV.1 and IV.2 present a summary of the results presented in Tables IV.3 and IV.4, to facilitate the analysis. In these graphs, we present with each color the BST and Average measures of solution quality, BST and Average, and the width represents the computational time spent, that is when thinner faster. With a simple look in Figures IV.1 and IV.2, one may verify everything that was said during the discussion of the results. Worth mentioning that the method from Genetic Algorithm presented by Fernandes et al. [2014] only the AVG solution was presented.

Looking at Table IV.3, one may notice that the proposed methods have the best mean of best solutions, the best mean of averages, the best time among all methods, and for four instances, it has new best solutions. The computational time, from the CS-ALNS-LB without reduction is approximately 25% faster than the CS+CPLEX running on the same machine and almost 20% faster than the BRKGA-LS. The method at least ties in 80% of the results presented, making the two presented versions of CS-ALNS-LB presented a viable option for the TSCFL. One may notice that our method obtains the minimum value in the best solutions for 24 out of the 25 instances. Is possible to notice that the CS-ALNS-LB without the reduction is the fastest of the methods and the solution of the two versions have the best means, but with reduction we obtain a better best solution for the last instance presented that already was a new best, despite this the method with reduction has an increase in time, although the method still faster than the others in literature.

In Class 3 instances, for which the mixed integer programming formulation failed to achieve optimum with the amount of available RAM used in Louzada et al. [2016], the CS-ALNS-LB obtained the best solutions in 4 out of 5 instances.

Table IV.4: Results obtained for the second set of instances (100 factories, 200 warehouses and 400 customers).

Instances			GA		CS+CPLEX			HEA/FA			BRKGA+LS			GRASPH			CS-ALNS-LB woRD			CS-ALNS-LB wRD		
Class	Id	LB	AVG	Time	BST	AVG	Time	BST	AVG	Time	BST	AVG	Time	BST	AVG	Time	BST	AVG	Time	BST	AVG	Time
	1	1475952	0.55	1268.12	0.10	0.30	384.79	0.29	0.33	1390.57	0.10	0.11	**265.79**	0.10	**0.10**	381.21	**0.09**	0.11	339.69	**0.09**	0.12	404.29
	2	1462736	1.01	1250.09	0.34	0.70	716.06	0.27	0.43	1569.57	**0.12**	0.13	547.62	0.12	**0.12**	**130.86**	0.12	0.20	231.41	**0.11**	0.21	281.07
1	3	1492163	0.34	1367.04	0.54	1.00	654.10	0.23	0.48	1535.84	**0.15**	0.17	**192.48**	0.15	**0.15**	281.03	0.16	0.20	266.18	**0.14**	0.20	323.47
	4	1459076	0.49	1285.78	0.24	0.49	740.44	0.25	0.28	1163.24	**0.22**	0.23	284.96	**0.22**	0.28	484.13	**0.22**	0.24	**240.00**	**0.22**	0.24	301.33
	5	1490742	0.67	1303.93	**0.11**	0.33	850.42	0.17	0.24	1555.82	0.12	0.12	515.29	0.12	0.12	**82.79**	**0.11**	0.12	192.28	**0.11**	**0.11**	323.15
	1	970908.5	0.89	675.48	**0.26**	0.52	989.39	0.63	0.70	979.27	0.27	0.28	398.44	0.27	0.36	584.20	**0.26**	0.30	**310.07**	**0.26**	0.31	381.94
	2	965908.5	0.74	662.96	**0.28**	0.46	668.55	0.47	0.56	941.41	0.29	0.29	458.38	**0.28**	0.34	559.24	**0.28**	0.33	**257.51**	**0.28**	0.31	371.92
2	3	975499.7	1.42	650.19	**0.14**	0.25	992.58	0.20	0.25	955.61	**0.14**	0.15	**89.55**	**0.14**	**0.14**	487.39	**0.14**	0.17	236.91	0.15	0.18	266.17
	4	973019.1	0.56	657.63	**0.28**	0.40	688.28	0.56	0.59	1123.11	**0.28**	**0.29**	594.11	0.35	0.41	592.81	0.29	0.32	**326.07**	**0.28**	0.31	622.56
	5	941567	1.12	646.23	**0.60**	0.65	858.06	0.86	0.96	1201.59	0.61	**0.61**	506.97	0.86	1.06	592.22	**0.60**	0.69	**283.63**	**0.60**	0.70	510.43
	1	5213566	**1.63**	617.24	**1.62**	**1.63**	1113.37	1.91	2.03	886.87	1.64	1.65	704.88	1.79	1.92	**598.74**	**1.62**	1.66	600.74	1.63	1.68	1274.40
	2	5191321	1.67	601.51	**1.65**	**1.65**	1312.48	1.96	1.99	1264.97	1.68	1.71	**234.85**	1.84	1.94	596.33	1.67	1.71	503.07	1.67	1.69	831.08
3	3	5145991	**1.58**	597.43	**1.57**	**1.58**	958.88	1.78	1.84	1236.61	1.63	1.63	**90.64**	1.77	1.84	594.41	1.58	1.61	491.88	**1.57**	1.63	922.62
	4	5225601	1.74	622.04	1.72	**1.73**	1033.46	1.98	2.01	1003.95	1.74	1.75	**313.32**	2.01	2.09	600.38	1.72	1.76	498.75	**1.71**	1.75	769.28
	5	5163182	1.72	629.84	**1.67**	**1.69**	1073.08	1.99	2.08	976.64	1.72	1.75	**409.97**	2.02	2.03	594.38	1.73	1.75	518.88	1.73	1.77	628.80
	1	1052172	0.82	577.91	**0.61**	0.87	1040.75	0.91	1.04	910.67	0.64	0.69	437.63	0.73	0.74	594.13	0.65	0.78	**313.52**	0.67	0.86	399.71
	2	1043553	0.93	560.18	0.83	0.88	852.22	0.82	0.89	697.12	0.68	0.74	496.97	0.77	0.85	593.65	**0.67**	**0.71**	**314.25**	**0.67**	0.72	366.50
4	3	1050683	1.88	584.40	**0.62**	**0.81**	677.12	1.00	1.29	909.24	0.73	1.04	566.19	1.20	1.77	592.49	0.78	1.12	**299.23**	0.89	1.28	363.30
	4	1044571	0.96	592.63	**0.74**	0.94	1099.22	1.36	1.62	826.32	0.81	0.88	573.03	0.98	1.01	594.38	0.80	0.90	**303.95**	0.75	**0.88**	364.91
	5	1053869	0.64	607.94	0.56	0.89	543.26	0.94	1.06	873.75	0.58	0.61	494.88	0.56	0.65	594.12	**0.52**	0.63	**295.09**	**0.52**	0.63	310.05
	1	5486098	0.48	706.40	**0.38**	**0.40**	801.30	0.61	0.62	1002.84	**0.39**	0.41	358.97	0.43	0.54	593.59	**0.38**	**0.40**	**326.89**	0.39	0.45	483.39
	2	5461680	0.47	683.08	0.39	0.44	849.41	0.41	0.44	1054.17	0.41	0.43	500.54	0.40	0.42	593.52	**0.38**	**0.41**	**285.26**	0.39	**0.41**	460.36
5	3	5425391	0.62	672.63	0.49	0.53	770.91	0.64	0.73	1421.97	0.42	0.45	555.96	0.59	0.59	594.61	**0.39**	0.48	**259.29**	**0.39**	0.51	571.17
	4	5494811	0.52	689.38	0.43	0.47	657.66	0.58	0.58	1123.96	0.43	0.44	363.07	0.50	0.51	593.16	**0.41**	**0.43**	**322.00**	**0.41**	**0.43**	484.96
	5	5442621	0.47	670.38	0.39	0.45	1323.85	0.43	0.47	1205.01	0.39	0.40	502.99	0.46	0.48	592.62	**0.38**	**0.40**	**262.75**	**0.38**	**0.40**	602.18
	Average		0.96	767.22	0.66	0.80	865.99	0.85	0.94	1112.40	0.65	**0.68**	418.30	0.75	0.82	523.86	**0.64**	0.70	331.17	**0.64**	0.71	504.76

[a] Fernandes et al. [2014]
[b] Guo et al. [2017]
[c] Biajoli et al. [2019]
AVG, BST in percentage and Time in seconds
[d] Gonzalez et al. [2019b]

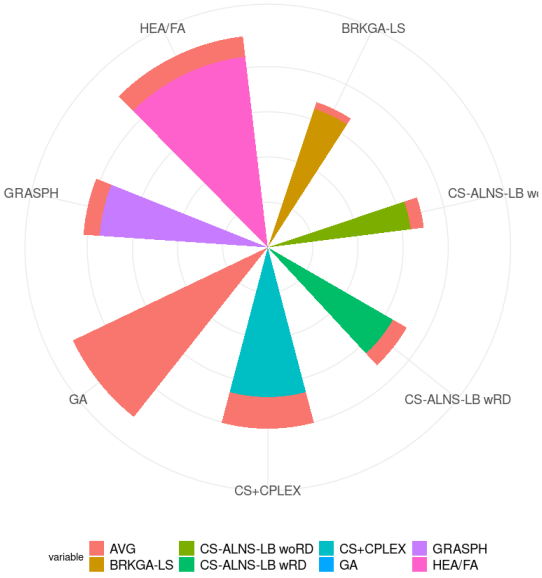

Figure IV.2: Summary of the Results for Instances of 100 Factories.

Table IV.4 shows that CS-ALNS-LB without Reduction is still faster than the other methods, the same difference of 20% is maintained in comparison of the fast method in the literature which was BRKGA-LS, and the mean of the best solutions is better than all of the presented methods. Although the BRKGA-LS surpasses the mean of average solutions of CS-ALNS-LB two versions by 0.02%, the CS-ALNS-LB outperformed the other methods. It is important to notice that the CS-ALNS-LB without reduction has achieved the best solutions for 16 out of 25 instances, and for two instances, a new minimum solution has been found. Another important remark is that BRKGA-LS uses a parallel approach to evaluate their population of solutions, with implicates that in a sequential version, BRKGA-LS will probably spend more computational time in order to achieve the presented results.

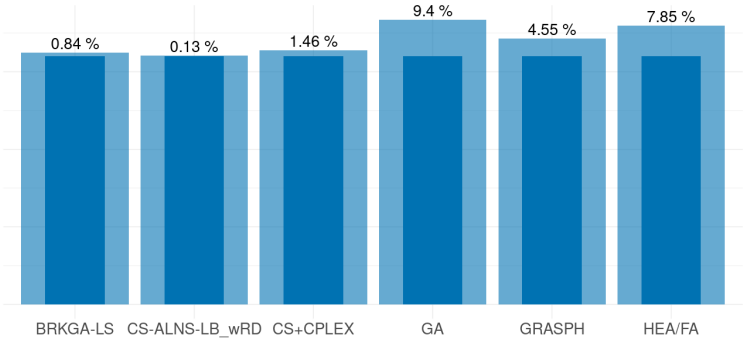

Figure IV.3: Comparison of the summation of the best results, presented in the literature, between CS-ALNS-LBwoRD and the other methods.

Analyzing Figure IV.3, one may verify the gain of the proposed method in relation to the best solutions reported in the literature, represented by the gap between the dark and light blue. Given the presented data, it is possible to verify that in the worst case (BRKGA-LS) the improvement

is 0.84%, while in the best case (GA) the improvement is 9.4%. From a practical point of view, improvements of 0.84% in their planning, as obtained by the proposed method, may implicate in a considerable improvement in the company's profit. The difference from the version with reduction is even closer but only the without reduction is compared with the others as it is the best solution.

After analyzing the results, one may conclude that the proposed CS-ALNS-LB is a viable option for finding high-quality solutions in low computational time for TSCFL. The next subsection will evaluate the other methodology which is the HBRKGA with and without the reduction in space.

IV.2 Computational Results - HBRKGA

In this section the same structure of evaluation is made, first we present the parameters used by the HBRKGA, then we present the the results and for last we show the improvement.

As the author wanna to control the time and obtain the best solution, parameters tested were obtained in a manual tuning to a satisfactorily degree. IRace package was not used by an author limitation of doesn't understanding how to use the multi-objective tuning. The parameters used are presented in IV.5.

Table IV.5: Values tested and selected for each parameter of the HBRKGA.

Parameters	Intervals	Selected Values
α	$\{0.60; 0.65; 0.70; 0.75; 0.80; 0.85; 0.90; \}$	0.75
p	$\{50; 60; 70; 80; 90\}$	50
pe	$\{0.2; 0.3; 0.4\}$	0.2
pm	$\{0.3; 0.4; 0.5\}$	0.3
ρ	$\{0.6; 0.7; 0.8\}$	0.7
gen	$\{80; 90; 100\}$	80

This parameters used are:

- α parameter of the constructive
- p size of the population
- pe percentage of the population in the elite set
- pm percentage of the population that will be applied the mutation process
- ρ probability of an elite passing its genes to the new offspring
- gen amount of generations used for the algorithm

Tables IV.6 and IV.7 using the same structure presented in Section IV.1.

Table IV.6: Results obtained for the first set of instances (50 factories, 100 warehouses and 200 customers) - HBRKGA.

Instances			GA		CS+CPLEX			HEA/FAc			BRKGA+LS			GRASPH			HBRKGAwoRD			HBRKGAwRD		
Class	Id	LB	AVG	Time	Best	AVG	Time	Best	AVG	Time	Best	AVG	Time	Best	AVG	Time	Best	AVG	Time	Best	AVG	Time
	1	721209.6	**0.13**	264.78	**0.13**	0.22	48.90	**0.13**	0.24	431.34	0.14	0.14	19.32	**0.13**	**0.13**	**4.50**	**0.13**	**0.13**	17.60	**0.13**	**0.13**	21.97
	2	730451.6	0.40	257.17	**0.23**	0.31	76.86	0.31	0.36	510.74	0.24	0.24	115.93	0.24	0.24	**21.34**	**0.23**	**0.23**	25.11	**0.23**	**0.23**	28.18
1	3	731885.3	0.24	263.35	**0.21**	0.29	51.94	0.22	0.24	327.60	0.22	0.22	39.29	0.22	0.22	**30.46**	**0.21**	**0.21**	91.04	**0.21**	**0.21**	108.35
	4	721515	0.81	242.93	1.19	1.41	96.62	0.54	0.59	472.15	**0.50**	0.51	116.29	**0.50**	**0.50**	29.08	**0.50**	**0.50**	**28.48**	**0.50**	**0.50**	34.60
	5	713633.8	0.82	251.79	**0.81**	0.88	**56.14**	0.86	0.97	480.59	**0.81**	**0.81**	59.10	**0.81**	**0.81**	160.01	**0.81**	**0.81**	86.16	**0.81**	**0.81**	113.84
	1	479860.2	2.69	144.39	**2.68**	3.27	27.69	2.74	2.82	239.25	2.69	2.69	16.84	2.69	2.69	383.27	**2.68**	**2.68**	**12.35**	**2.68**	**2.68**	14.85
	2	483072.2	**2.30**	144.16	**2.30**	2.62	81.36	2.34	2.41	206.37	**2.30**	2.31	60.58	**2.30**	2.34	368.58	**2.30**	**2.30**	63.53	**2.30**	**2.30**	**17.32**
2	3	486018.5	2.14	150.60	**1.86**	2.03	**30.82**	1.87	1.89	173.38	1.87	1.87	117.20	1.88	1.94	590.94	**1.86**	**1.86**	57.74	**1.86**	**1.86**	52.45
	4	482374.6	2.04	142.25	**2.01**	**2.01**	83.02	2.07	2.12	168.04	2.02	2.05	59.59	2.02	2.02	365.57	**2.01**	**2.01**	49.86	**2.01**	**2.01**	**16.51**
	5	474803.3	3.14	126.08	**3.12**	3.39	36.98	**3.12**	**3.12**	176.79	**3.12**	**3.12**	**10.84**	**3.12**	**3.12**	590.87	**3.12**	**3.12**	13.29	**3.12**	**3.12**	15.85
	1	2608800	**3.07**	125.90	**3.07**	**3.07**	**19.89**	3.14	3.30	121.36	3.11	3.11	126.94	3.22	3.30	596.03	**3.07**	3.11	61.56	**3.07**	3.12	65.63
	2	2616252	3.12	130.22	**3.10**	**3.10**	73.09	3.30	3.36	120.92	3.17	3.17	**40.26**	3.37	3.39	594.73	**3.10**	3.21	67.88	3.13	3.18	63.74
3	3	2598277	3.11	123.56	**3.09**	**3.10**	**52.98**	3.30	3.39	167.70	3.10	**3.10**	72.94	3.23	3.32	591.33	**3.09**	3.17	71.75	**3.09**	3.18	67.43
	4	2612534	3.07	107.73	**3.05**	**3.05**	45.21	3.18	3.22	154.29	3.10	3.10	**8.25**	3.18	3.29	593.68	3.13	3.15	60.83	3.12	3.17	65.27
	5	2568856	**3.01**	110.36	**3.01**	**3.01**	47.45	3.17	3.19	163.91	3.13	3.13	**40.23**	3.14	3.22	593.27	3.02	3.03	66.45	**3.01**	3.03	68.10
	1	525294.1	**3.14**	138.25	**3.14**	3.60	89.47	3.36	3.54	224.71	**3.14**	3.22	58.44	3.29	3.29	591.54	**3.14**	**3.14**	30.70	**3.14**	**3.14**	**24.63**
	2	526911.7	**2.33**	139.83	**2.43**	2.71	102.38	2.74	2.92	206.66	**2.43**	2.51	80.98	2.65	2.80	592.19	**2.43**	2.45	68.26	**2.43**	2.43	**45.52**
4	3	532592.3	2.66	144.88	2.41	2.44	118.38	2.59	2.62	256.30	2.33	2.42	110.29	2.45	2.92	591.35	**2.27**	2.40	100.76	**2.27**	**2.37**	**70.90**
	4	529372	2.53	127.30	**2.35**	2.66	133.69	2.63	2.81	271.29	2.44	2.51	74.85	2.36	2.50	591.31	**2.35**	2.36	103.25	**2.35**	**2.35**	**44.00**
	5	521470.1	3.13	120.27	3.15	3.53	115.67	3.18	3.18	188.88	3.14	3.16	39.57	3.15	3.23	388.72	**3.12**	**3.12**	44.55	**3.12**	**3.12**	**35.80**
	1	2743547	1.20	164.42	1.19	1.19	157.20	**1.16**	1.20	206.49	1.18	1.22	120.41	1.24	1.31	591.33	**1.16**	1.18	64.92	**1.16**	**1.17**	**64.78**
	2	2752021	**1.07**	156.71	1.08	1.11	89.13	1.11	1.16	269.19	**1.07**	1.09	**35.71**	1.15	1.17	591.31	**1.07**	**1.07**	75.68	**1.07**	**1.07**	75.33
5	3	2737769	**1.10**	191.60	**1.09**	**1.10**	126.33	1.22	1.28	210.10	1.13	1.15	130.43	1.29	1.30	591.78	**1.09**	1.17	**66.25**	**1.09**	1.13	68.00
	4	2748216	**1.07**	136.87	**1.05**	1.12	149.59	1.10	1.13	210.73	1.10	1.10	151.61	1.06	**1.07**	591.67	**1.05**	1.10	**69.07**	**1.05**	**1.07**	72.66
	5	2702350	1.25	145.07	1.24	**1.24**	**54.96**	1.31	1.34	188.84	1.26	1.27	74.20	1.29	1.34	592.50	**1.22**	1.25	72.84	**1.22**	**1.24**	67.59
Average			1.98	162.02	1.96	2.10	78.63	2.03	2.10	245.91	1.95	1.97	71.20	2.00	2.06	449.09	**1.93**	1.95	56.86	**1.93**	**1.94**	**52.93**

[a] Fernandes et al. [2014] [b] Guo et al. [2017]
[c] Biajoli et al. [2019] AVG, BST in percentage and Time in seconds
[d] Gonzalez et al. [2019b]

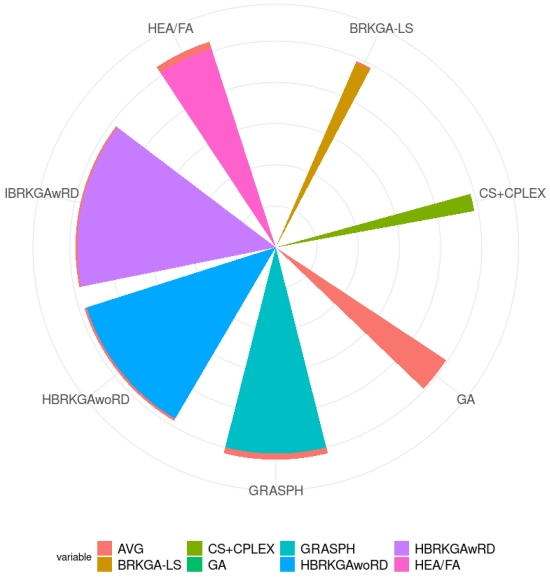

Figure IV.4: Summary of the Results for Instances of 50 Factories.

Analyzing Table IV.6 we note a behavior different from the one analyzed in Table IV.3, here the reduction in the search space was effective not only for the solution quality, but also in the computational time giving a result that is the best when compared to all the other methods in the literature. Maintaining the proportion of computational time spent in 20% better than the BRKGA-LS, worth mentioning one more time that our method is a sequential version and still the fastest. Another thing that is valid to mention is that in HBRKGA we obtained for three out of five in the third class, which was a problem to the CS-ALNS-LB.

Table IV.7: Results obtained for the second set of instances (100 factories, 200 warehouses and 400 customers).

Instances			GA		CS+CPLEX			HEA/FA			BRKGA+LS			GRASPH			HBRKGA woRD			HBRKGA wRD		
Class	Id	LB	AVG	Time	BST	AVG	Time	BST	AVG	Time	BST	AVG	Time	BST	AVG	Time	BST	AVG	Time	BST	AVG	Time
1	1	1475952	0.55	1268.12	0.10	0.30	384.79	0.29	0.33	1390.57	0.10	0.11	**265.79**	0.10	0.10	381.21	**0.09**	0.10	927.11	**0.09**	**0.09**	464.64
	2	1462736	1.01	1250.09	0.34	0.70	716.06	0.27	0.43	1569.57	0.12	0.13	547.62	0.12	0.12	**130.86**	**0.11**	**0.11**	185.93	**0.11**	**0.11**	211.66
	3	1492163	0.34	1367.04	0.54	1.00	654.10	0.23	0.48	1535.84	0.15	0.17	**192.48**	0.15	**0.15**	281.03	**0.14**	**0.15**	466.20	**0.14**	**0.15**	503.93
	4	1459076	0.49	1285.78	0.24	0.49	740.44	0.25	0.28	1163.24	**0.22**	0.23	284.96	**0.22**	0.28	484.13	**0.22**	**0.22**	597.05	**0.22**	**0.22**	**225.92**
	5	1490742	0.67	1303.93	**0.11**	0.33	850.42	0.17	0.24	1555.82	0.12	0.12	515.29	0.12	0.12	**82.79**	**0.11**	**0.11**	260.53	**0.11**	**0.11**	267.1
2	1	970908.5	0.89	675.48	**0.26**	0.52	989.39	0.63	0.70	979.27	0.27	0.28	**398.44**	0.27	0.36	584.20	**0.26**	0.27	468.02	**0.26**	**0.26**	548.76
	2	965908.5	0.74	662.96	**0.28**	0.46	668.55	0.47	0.56	941.41	0.29	0.29	458.38	**0.28**	0.34	559.24	**0.28**	**0.28**	349.86	**0.28**	**0.28**	**206.42**
	3	975499.7	1.42	650.19	**0.14**	0.25	992.58	0.20	0.25	955.61	**0.14**	0.15	**89.55**	**0.14**	**0.14**	487.39	**0.14**	**0.14**	403.85	**0.14**	**0.14**	514.52
	4	973019.1	0.56	657.63	**0.28**	0.40	688.28	0.56	0.59	1123.11	**0.28**	**0.29**	594.11	0.35	0.41	592.81	**0.28**	0.32	**463.45**	**0.28**	0.3	495.35
	5	941567	1.12	646.23	**0.60**	0.65	858.06	0.86	0.96	1201.59	0.61	**0.61**	506.97	0.86	1.06	592.22	**0.60**	0.70	**421.81**	**0.60**	0.62	452.98
3	1	5213566	**1.63**	617.24	**1.62**	**1.63**	1113.37	1.91	2.03	886.87	1.64	1.65	704.88	1.79	1.92	**598.74**	1.65	1.67	1182.05	1.63	1.66	1208.8
	2	5191321	1.67	601.51	**1.65**	**1.65**	1312.48	1.96	1.99	1264.97	1.68	1.71	**234.85**	1.84	1.94	596.33	1.66	1.70	1623.16	**1.65**	1.68	1331.01
	3	5145991	**1.58**	597.43	**1.57**	**1.58**	958.88	1.78	1.84	1236.61	1.63	1.63	**90.64**	1.77	1.84	594.41	1.58	1.61	1473.17	1.58	1.6	1126.57
	4	5225601	1.74	622.04	**1.72**	**1.73**	1033.46	1.98	2.01	1003.95	1.74	1.75	**313.32**	2.01	2.09	600.38	1.73	1.76	1205.52	1.73	1.78	1311.34
	5	5163182	1.72	629.84	**1.67**	**1.69**	1073.08	1.99	2.08	976.64	1.72	1.75	**409.97**	2.02	2.03	594.38	1.69	1.76	1453.98	**1.67**	1.73	2578.49
4	1	1052172	0.82	577.91	0.61	0.87	1040.75	0.91	1.04	910.67	0.64	0.69	**437.63**	0.73	0.74	594.13	0.60	0.64	444.92	**0.58**	**0.63**	881.31
	2	1043553	0.93	560.18	0.83	0.88	852.22	0.82	0.89	697.12	0.68	0.74	496.97	0.77	0.85	593.65	**0.67**	0.68	**461.42**	**0.67**	**0.67**	832.7
	3	1050683	1.88	584.40	**0.62**	**0.81**	677.12	1.00	1.29	909.24	0.73	1.04	566.19	1.20	1.77	592.49	0.74	0.90	**381.89**	0.69	0.88	640.11
	4	1044571	0.96	592.63	**0.74**	0.94	1099.22	1.36	1.62	826.32	0.81	0.88	**573.03**	0.98	1.01	594.38	**0.74**	0.76	770.15	**0.74**	**0.74**	1104.12
	5	1053869	0.64	607.94	0.56	0.89	543.26	0.94	1.06	873.75	0.58	0.61	494.88	0.56	0.65	594.12	**0.52**	0.54	**377.55**	**0.52**	**0.52**	453.17
5	1	5486098	0.48	706.40	**0.38**	0.40	801.30	0.61	0.62	1002.84	0.39	0.41	**358.97**	0.43	0.54	593.59	**0.38**	0.40	687.61	**0.38**	**0.39**	781.56
	2	5461680	0.47	683.08	0.39	0.44	849.41	0.41	0.44	1054.17	0.41	0.43	**500.54**	0.40	0.42	593.52	0.39	**0.39**	563.30	**0.38**	**0.39**	1197.15
	3	5425391	0.62	672.63	0.49	0.53	770.91	0.64	0.73	1421.97	0.42	0.45	555.96	0.59	0.59	594.61	**0.39**	**0.42**	**546.97**	0.40	**0.42**	647.22
	4	5494811	0.52	689.38	0.43	0.47	657.66	0.58	0.58	1123.96	0.43	0.44	**363.07**	0.50	0.51	593.16	**0.40**	**0.42**	458.92	**0.40**	**0.42**	507.92
	5	5442621	0.47	670.38	0.39	0.45	1323.85	0.43	0.47	1205.01	0.39	0.40	502.99	0.46	0.48	592.62	0.39	0.40	**403.65**	**0.38**	**0.39**	683.12
Average			0.96	767.22	0.66	0.80	865.99	0.85	0.94	1112.40	0.65	0.68	**418.30**	0.75	0.82	523.86	**0.63**	0.66	663.12	**0.63**	**0.65**	767.03

[a] Fernandes et al. [2014] [b] Guo et al. [2017]
[c] Biajoli et al. [2019] AVG, BST in percentage and Time in seconds
[d] Gonzalez et al. [2019b]

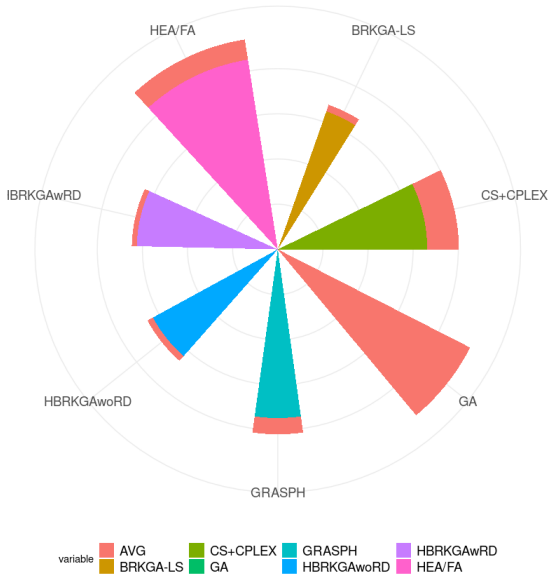

Figure IV.5: Summary of the Results for Instances of 50 Factories.

Now in the analyses of Table IV.7, one may notice that the HBRKGA still maintains its solution quality, giving the best result for 20 out of 25, and best average solution. Although the HBRKGA is almost two times slower than the BRKGA-LS, the nature of the problem seems to not require the computational for problem solving because the facility location problem tend to have a static nature for a long period of time, so HBRKGA shows itself to be a good option for solution quality in the problem.

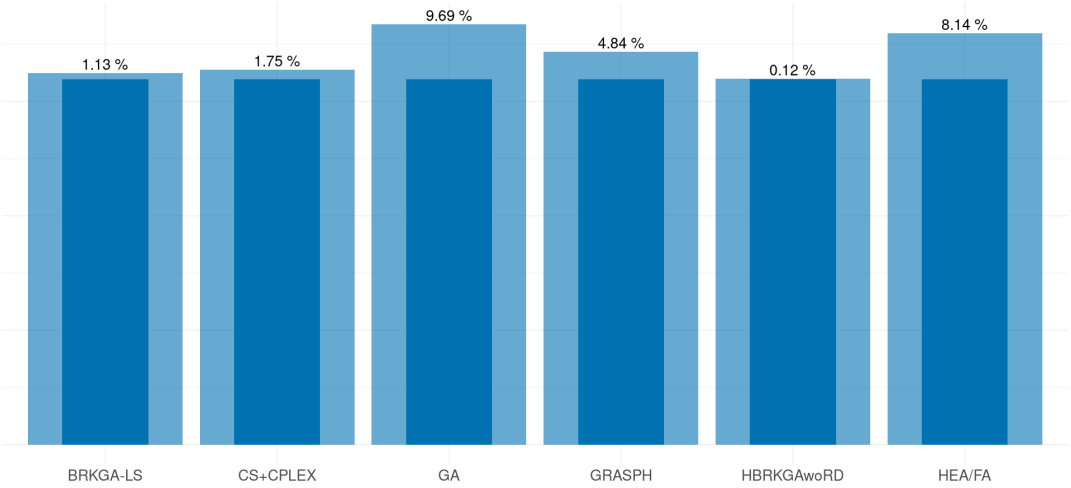

Figure IV.6: Comparison of the summation of the best results, presented in the literature, between HBRKGAwRD and the other methods..

HBRKGAwRD with reduction is the best version we compare it to the others methods and the distance from the bests is even better than the CS-ALNS-LB 1.13% when compared to the 0.84% worth mentioning that the method is outperforming all methods in the solution quality comparison.

This dissertation have presented the two methods and its results with two methodologies that

have good results in solution quality and time, in order to do a comparison and confirm that the presented methods are a good option the next section will make an statistical comparison between the two methods that are the state-of-art from the literature.

IV.3 Comparing CS-ALNS-LB and HBRKGA

In order to verify whether or not the differences of values obtained by the state-of-art strategies and the proposed methods are statistically significant, a hypothesis test was executed. In addition to the tables of computational experiments for the proposed method, the hypothesis test was performed based on the results of the executions sent to us by the authors of Biajoli et al. [2019] and the new execution of the method CS+CPLEX in our machine. As it was not possible to prove that the data fit a normal distribution, the Willcoxon-Mann-Whitney [Conover, 1999] non-parametric test was used. This test can reject the null hypothesis of θ level, with the probability $(1 - \theta \times 100\%)$. In this dissertation proposal we considered $\theta = 0.05$.

- Null Hypothesis (H0): There are no significant difference between our methods and the other methods
- Alternative Hypothesis (H1): There are significant difference between our methods and the other methods

Table IV.8: Number of times the Null Hypothesis was rejected, in the hypothesis test ,when comparing CS-ALNS-LB with or without reduction against CS+CPLEX, BRKGA-LS and HBRKGA with and without reduction.

Instance groups	CS-ALNS-LB woRD			
	w/ CS+CPLEX	w/ BRKGA-LS	w/ HBRKGA woRD	w/ HBRKGA wRD
50	19/25	17/25	3/25	1/25
100	16/25	5/25	9/25	16/25
Instance groups	CS-ALNS-LB wRD			
	w/ CS+CPLEX	w/ BRKGA-LS	w/ HBRKGA woRD	w/ HBRKGA wRD
50	17/25	14/25	3/25	7/25
100	15/25	10/25	15/25	17/25

Our tests, presented in Table IV.8, shown the number of instances for which the Null Hypothesis was rejected. For instances with 50 factories, it is clear that the CS-ALNS-LB is different from the other methods since, for more than 60% of the tested instances, the Null Hypothesis was rejected in comparison to the literature methods. Analyzing the difference between the HBRKGA, was not possible to verify a significant difference for the instances of 50 factories. Considering the results for instances with 100 factories, one may verify that when comparing the CS-ALNS-LB with the CS+CPLEX, for 16 out of 25 instances and 17 out of 25,in the case with the Reduction space, the

Null Hypothesis was rejected, proving that they are different even the HBRKGA has a significant difference to CS-ALNS-LB. Now, still in the set of instances with 100 factories, when comparing the CS-ALNS-LB with BRKGA-LS, the Null Hypothesis is rejected just for 5 out of 25 instances, which may lead us to conclude that there is no significant difference between both algorithms for this set of instances. This could have happened thanks to the fact that the mean of averages solution of the BRKGA-LS is lower than the one obtained by CS-ALNS-LB. This was a relevant analysis to apply the HBRKGA to the problem where we want to find better quality solutions, regardless the loss in time. **Comparação CS-ALNS-LB woRD e wRD**

Table IV.9: Number of times the Null Hypothesis was rejected, in the hypothesis test ,when comparing HBRKGA with or without reduction against CS+CPLEX, BRKGA-LS and CS-ALNS-LB with and without reduction.

Instance groups	HBRKGA woRD			
	w/ CS+CPLEX	w/ BRKGA-LS	w/ CS-ALNS-LB woRD	w/ CS-ALNS-LB wRD
50	20/25	18/25	1/25	3/25
100	20/25	15/25	9/25	15/25
Instance groups	HBRKGA wRD			
	w/ CS+CPLEX	w/ BRKGA-LS	w/ CS-ALNS-LB woRD	w/ CS-ALNS-LB wRD
50	21/25	20/25	3/25	7/25
100	20/25	18/25	16/25	17/25

Table IV.9, shows that the the improvement in the average quality of solutions was relevant to propose this new methodology to the problem. The Null Hypothesis was denied for 84% in the instances of 50 factories and for 72% in the 100 factories, when compared to the BRKGA-LS. This shows a stability of HBRKGA in finding solutions of quality, and can show that our methodology have significant difference between the others in the literature.Next chapter presents the conclusions for the two methods and presents future works.

Chapter V Conclusion

V.1 Conclusions

This work presented a hybridization of CS and ALNS metaheuristics was not addressed by the literature. In addition, CS-ALNS was also combined with a Local Branching technique, resulting in the proposed CS-ALNS-LB matheuristic, we also presented the HBRKGA, although this combination of components is not new, as we can find BRKGA with local search and with local branching in the literature, was the first time was applied to this problem and shows that our method outperformed the others in the task of finding quality solutions.

The performance of the two proposed methods was compared with the most relevant approaches in the literature and a structure used to reduce the space was proposed. The obtained results showed that the HBRKGA outperformed the other approaches both in the quality of the solution found, as well as in the computational time spent. Regarding the quality of the solutions found, for the instances with 50 factories, CS-ALNS-LB and HBRKGA found mean results for both best and average solutions better than all the other analyzed algorithms and for the instances of 100 factories HBRKGA still found better average and best solutions. Besides, in terms of computational time, HBRKGA was, on average, 2 times slower than state-of-the-art method BRKGA-LS, when the application needs for better quality solution HBRKGA can be applied as stable best solutions can be found for instances. For the instances with 100 factories, despite CS-ALNS-LB having lost performance, it still obtained mean best solution better than those from all analyzed algorithms, while the mean average solution was worse just for BRKGA-LS. In terms of computational time, for the instances of 100 factories, CS-ALNS-LB was, on average, 1.20 times faster than state-of-the-art methods (CS-CPLEX and BRKGA-LS), showing that when performance is needed it can be applied. Also, one should not forget that BRKGA-LS is a parallel metaheuristic, so the reduction in computational time is even greater if one consider it a sequential approach, which could make the HBRKGA a method valid to solutions and computational time. In addition to the numerical results, a statistical analysis was presented, showing that the CS-ALNS-LB and HBRKGA differs from other literature approaches not only by its randomness.

With the dissertation, we demonstrated the knowledge developed for the TSCFL, showing that the proposed components are robust for the application studied and that it can be used

as the structure to explore other problems. As future works its possible to implement and test decomposition techniques of exact methods to test, how it can improve the methods of hybridization tested in this dissertation. Implement the heuristics of reinforcement learning can be a way of testing new components that are showing promising structures in other problems. Construct this dissertation proposed applied to more complex scenarios and real applications like the one presented by Guimarães et al. [2021].

Bibliography

A. C. M. de Oliveira, A. A. Chaves, and L. A. N. Lorena. Clustering search. *Pesquisa Operacional*, 33:105 – 121, 04 2013. ISSN 0101-7438. , 15, 21, 29, 32

D. R. M. Fernandes, C. Rocha, D. Aloise, G. M. Ribeiro, E. M. Santos, and A. Silva. A simple and effective genetic algorithm for the two-stage capacitated facility location problem. *Computers & Industrial Engineering*, 75:200–208, 2014. , 14, 17, 36, 38, 39, 40, 43, 45

Peng Guo, W. Cheng, and Y. Wang. Hybrid evolutionary algorithm with extreme machine learning fitness function evaluation for two-stage capacitated facility location problems. *Expert Systems With Applications*, 71:57–68, 2017. , 14, 36, 37, 38, 40, 43, 45

F. L. Biajoli, A. A. Chaves, and L. A. N. Lorena. A biased random-key genetic algorithm for the two-stage capacitated facility location problem. *Expert Systems With Applications*, 115:418–426, 2019. , 14, 36, 38, 40, 43, 45, 47

P. H. Gonzalez, Souto G., et al. Grasp híbrido para resolução do problema de localização de facilidades capacitadas em dois níveis. *Simpósio Brasileiro de Pesquisa Operacional - SBPO*, pages 1–11, 2019b. , 14, 36, 38, 40, 43, 45

R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman. Intelligent manufacturing in the context of industry 4.0: a review. *Engineering*, 3(5):616–630, 2017. 13

V. Guimarães, G. C. Skroder, G. M. , and P. H. González. Strategic planning of freight transportation to support smart cities design: The Brazilian soybean case. *Revista Facultad de IngenierÃa Universidad de Antioquia*, pages 104 – 116, 03 2021. ISSN 0120-6230. URL `http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-62302021000100104&nrm=iso`. 13, 50

A. M. Geoffrion and G. W. Graves. Multicommodity distribution system design by benders decomposition. *Management Science*, 20(5):822–844, 1974. 13

S. Tragantalerngsak, J. Holt, and M. Ro"nnqvist. Lagrangian heuristics for the two-echelon, single-source, capacitated facility location problem. *European Journal of Operational Research*, 102(3): 611–625, 1997. ISSN 0377-2217. 13

H. Pirkul and J. Vaidyanathan. A multi-commodity, multi-plant, capacitated facility location problem: formulation and efficient heuristic solution. *Computers & Operations Research*, 25(10): 869–878, 1998. ISSN 0305-0548. 13

G. R. Mauri, F. L. Biajoli, R. L. Rabello, A. A. Chaves, G. M. Ribeiro, and L. A. N. Lorena. Hybrid metaheuristics to solve a multiproduct two-stage capacitated facility location problem. *International Transactions in Operational Research*, n/a(n/a), 2021. 13

T. Wu, Z. Yang, F. Chu, and Z. Zhou. A lagrangean relaxation approach for a two-stage capacitated facility location problem with choice of depot size. In *2015 IEEE 12th International Conference on Networking, Sensing and Control*, pages 39–44. IEEE, 2015. 14

Z. Yang, H. Chen, F. Chu, and N. Wang. An effective hybrid approach to the two-stage capacitated facility location problem. *European Journal of Operational Research*, 275(2):467–480, 2019. 14

A. Klose. A lagrangian relax-and-cut approach for the two-stage capacitated. *European Journal of Operational Research*, 126:185–198, 2000. 14

S. Tragantalerngsak, J. Holt, and M. Ronnqvist. An exact method for the two-echelon, single-source, capacitated facility location problem. *European Journal of Operational Research*, 123: 473–489, 2000. 14

I. Litvinchev and E. L. Ozuna. Lagrangian bounds and a heuristic for the two-stage capacitated facility location problem. *International Journal of Energy Optimization and Engineering*, 1:59–71, 2012. 14, 16

A. Klose and A. Drexl. Facility location models for distribution system design. *European Journal of Operational Research*, 162:4–29, 2005. 14

R. R. Louzada, G. R. Mauri, and G. M. Ribeiro. Método heurístico híbrido para resolução do problema de localização de facilidades capacitadas em dois níveis. In *Simpósio Brasileiro de Pesquisa Operacional - SBPO*, pages 2460–2471, 2016. 14, 16, 19, 24, 37, 39

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, 40(4):455–472, 2006. 15, 21, 25

M. Fischetti and A. Lodi. Local branching. *Mathematical Programming*, 98:23–47, 2003. 15, 21, 28

M. Keskin and Çatay B. A matheuristic method for the electric vehicle routing problem with time windows and fast chargers. *Computers & Operations Research*, 100:172 – 188, 2018. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2018.06.019. 15

JP. F. Gonçalves and M. G. C. Resende. Biased random-key genetic algorithms for combinatorial optimization. *Journal of Heuristics*, 17(5):487–525, 2011. 15, 32

J. Orlin, S. Plotkin, and É. Tardos. Polynomial dual network simplex algorithms. *Mathematical Programming*, 60, 01 2001. doi: 10.1007/BF01580615. 20

R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows*. Cambridge, Mass.: Alfred P. Sloan School of Management, Massachusetts . . . , 1988. 21

P. H. V. Penna, A. Subramanian, and L. S. Ochi. An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19(2):201–232, 2013. 24

P. H. Gonzalez, L. Simonetti, P. Michelon, C. Martinhon, and E. Santos. A variable fixing heuristic with local branching for the fixed charge uncapacitated network design problem with user-optimal flow. *Computers & Operations Research*, 76:134–146, 2016. 28

P. H. Gonzalez and J. Brandão. A biased random key genetic algorithm to solve the transmission expansion planning problem with re-design. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–7. IEEE, 2018. 28

A. C. M. Oliveira and L. A. N. Lorena. Hybrid evolutionary algorithms and clustering search. In *HYBRID evolutionary algorithms*, pages 77–99. Springer, 2007. 29

A.A. Chaves, L.A.N. Lorena, E.L.F. Senne, and Resende M.G.C. Hybrid method with cs and brkga applied to the minimization of tool switches problem. *Computers & Operations Research*, 67:174 – 183, 2016. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2015.10.009. 29

R. A. Rosa, A. M. Machado, G. M. Ribeiro, and G. R. Mauri. A mathematical model and a clustering search metaheuristic for planning the helicopter transportation of employees to the production platforms of oil and gas. *Computers & Industrial Engineering*, 101:303–312, 2016. 29

P. H. González, G. Clímaco, G. R. Mauri, B. S. Vieira, G. M. Ribeiro, R. D. Orrico Filho, Luidi Simonetti, L. R. Perim, and I. C. S. Hoffmann. New approaches for the traffic counting location problem. *Expert systems with applications*, 132:189–198, 2019a. 29

G. Laporte, R. Musmano, and F. Vocaturo. An adaptive large neighborhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44(1): 125–135, 2010. 30

G. M. Ribeiro, G. Laporte, and G. R. Mauri. A comparison of three metaheuristics for the workover rig routing problem. *European Journal of Operational Research*, 220(1):28—-36, 2012. 30

G. R. Mauri, G. M. Ribeiro, L. A. N. Lorena, and G. Laporte. An adaptive large neighborhood search for the discrete and continuous berth allocation problem. *Computers & Operations Research*, 70:140—-154, 2016. 30

A. Kiefer, R. F. Hartl, and A. Schnell. Adaptive large neighborhood search for the curriculum-based course timetabling problem. *Annals of Operations Research*, 252(2):255—282, 2017. 30

J. C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA journal on computing*, 6(2):154–160, 1994. 32

López-Ibáñez Manuel, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, and T. Stützle. The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58, 2016. 37

W. J. Conover. *Practical nonparametric statistics*. Wiley series in probability and statistics. Wiley, New York, NY [u.a.], 3. ed edition, 1999. ISBN 0471160687. URL `http://gso.gbv.de/DB=2.1/CMD?ACT=SRCHA&SRT=YOP&IKT=1016&TRM=ppn+24551600X&sourceid=fbw_bibsonomy`. 47