algorithms

Article

# Solution Algorithms for the Capacitated Location Tree Problem with Interconnections

**Nidia Mendoza-Andrade [1], Efrain Ruiz-y-Ruiz [2,*] and Suemi Rodriguez-Romo [3]**

[1] Facultad de Estudios Superiores Cuautitlán, UNAM Universidad Autónoma de México, Ciudad de México 04510, Mexico; nidiama@comunidad.unam.mx
[2] TECNM Tecnológico Nacional de México, Instituto Tecnológico de Saltillo, Saltillo 25280, Mexico
[3] UNAM Facultad de Estudios Superiores Cuautitlán, Cuatitlán Izcalli 54740, Mexico; suemi@unam.mx
* Correspondence: hector.ry@saltillo.tecnm.mx

**Abstract:** This paper addresses the Capacitated Location Tree Problem with Interconnections, a new combinatorial optimization problem with applications in network design. In this problem, the required facilities picked from a set of potential facilities must be opened to serve customers using a tree-shaped network. Costs and capacities are associated with the opening of facilities and the establishment of network links. Customers have a given demand that must be satisfied while respecting the facilities and link capacities. The problem aims to minimize the total cost of designing a distribution network while considering facility opening costs, demand satisfaction, capacity constraints, and the creation of interconnections to enhance network resilience. A valid mixed-integer programming was proposed and an exact solution method based on the formulation was used to solve small- and medium-sized instances. To solve larger instances two metaheuristic approaches were used. A specific decoder procedure for the metaheuristic solution approaches was also proposed and used to help find solutions, especially for large instances. Computational experiments and results using the three solution approaches are also presented. Finally, a case study on the design of electrical transportation systems was presented and solved.

**Keywords:** Capacitated Location Tree Problem; location; network design; BRKGA; PSO

check for updates

Academic Editor: Tatsuya Akutsu

Received: 10 December 2024
Revised: 8 January 2025
Accepted: 13 January 2025
Published: 17 January 2025

**Citation:** Mendoza-Andrade, N.; Ruiz-y-Ruiz, E.; Rodríguez-Romo, S. Solution Algorithms for the Capacitated Location Tree Problem with Interconnections. *Algorithms* **2025**, *18*, 50. https://doi.org/10.3390/a18010050

## 1. Introduction

In the intricate fabric of human history, the undulating currents of societal mobility have woven a complex narrative closely tied to the imperatives of commerce, sparking the birth and evolution of sophisticated transportation systems. As the population burgeons, the need for increased mobility propels the adoption of electric transportation systems—subways, electric trains, and cable cars—with each standing as a testament to innovation and progress. Within this evolving landscape, the pivotal consideration of electrical consumption emerges as the keystone, dictating the efficiency and efficacy of these burgeoning transit modes [1].

The escalating demands of a growing population have triggered an unprecedented demand for electric transportation services, compelling the evolution of systems capable of seamlessly delivering the required electrical power. The expansive growth of metro networks and the surging number of commuters at each station underscore the paramount importance of traction substations. These substations, acting as sentinels of power, are crucial in supplying the essential energy needed to maintain the pinnacle of service quality amidst rising demand [2].

Navigating this intricate scenario, an innovative algorithm takes center stage, poised to deftly address the complex challenge of strategically placing traction substations and managing their intricate interconnections. At its core, this algorithm fervently pursues the cardinal objective of minimizing the proliferation of necessary traction substations. Executed with meticulous precision, this endeavor is dedicated to optimizing energy efficiency and fortifying infrastructural integrity, constituting a cornerstone contribution to achieving the sustainability and operational excellence of burgeoning electric transportation systems [3].

Integral to this paradigm is the dynamic presence of traction current converter plants, serving as linchpins in the power supply network. These converter plants assume two principal forms: decentralized and centralized. In the decentralized model, a single plant directly supplies the overhead lines or third rail of the traction system without an intermediary feed into a traction current distribution network. Conversely, centralized converter plants play a dual role, supplying the traction power network while directly providing power to the overhead lines or third rail. This nuanced addition accentuates the multifaceted nature of the infrastructural landscape, where the strategic placement of traction substations harmonizes with the dual modality of traction current converter plants, orchestrating an intricate ballet of power distribution and efficiency in the realm of electric transportation [4].

This paper introduces the Capacitated Location Tree Problem with Interconnections (CLTPI). The problem arises from the need to optimally design electric transportation networks capable of partially functioning during disruptions. In the CLTPI, a subset of capacitated facilities (electrical substations) from a set of potential locations must be opened (built) to service a set of clients (traction substations) with a specific demand. Clients are serviced using a capacitated network that has to be established. There are costs associated with opening any of the potential facilities, as well as with the topological design of the network.

A formulation and two metaheuristic solution methods are proposed and used to solve a set of benchmark test instances from the literature. The benchmark instances are derived from the Location Routing problem, which is closely related to the CLTPI. Computational experiments and the results obtained from the three proposed solution methods are presented and discussed.

Section 2 presents a relevant literature review related to the CLTPI. Section 3 describes the problem and proposes a mixed-integer programming formulation. Three solution approaches are presented in Section 4. Computational experiments and their results are presented in Section 5, while the case study and its outcome are discussed in Section 6. Finally, the most relevant findings and potential future research are presented in Section 7.

## 2. Literature Review

Network design problems have been widely studied in the literature. The minimum spanning tree (MST) can be considered the fundamental problem for network design that can be solved using polynomial–time algorithms like the ones proposed by [5,6]. The MST considers a set of vertices that have to be connected through a network so that a commodity (electricity, information, water, etc.) can flow from any origin vertex to a destination one. When capacities restrict the flow through the network's links, the capacitated minimum spanning tree (CMST) arises [7]. By including such constraints, the problem becomes of the type NP-Hard, as Papadimitriou showed in [8].

On the other hand, location problems have also been extensively studied in the literature. The most basic location problem is the $p$-median problem or uncapacitated facilities location problem (UFLP) [9]. In this problem, $p$ facilities have to be open to serve a set of customers. In the capacitated version of the problem (capacitated facility location problem) [10], capacities are considered for plants, and each customer has a predefined demand to be delivered.

The CLTPI can be considered a combination of the CMST and CFLP problems with additional constraints (interconnections). One of the most related research works to the CLPTI is by Araóz et al. [11], which presents three location-routing problems (LRPs) defined over tree-shaped graphs. LRPs have been widely studied in the literature. Interesting reviews of LRPs can be found in [12,13]. The authors propose various LRPs defined over a tree-shaped road network (graph). The authors present several variants of the problem depending on the characteristics of customers' demand. In all the proposed variants, the best location has to be selected among a set of facilities at the graph's vertices, and customers are assigned to one selected facility. Finally, closed routes are designed using open facilities as depots while minimizing the total cost. The authors propose solution methods that first determine which facilities are open and assign the customers to such facilities. The optimal routing is then obtained by decomposing it into smaller subproblems associated with each open facility. Since the problem is defined over a tree-shaped graph, edges (the authors used a non-directed graph) are traversed precisely twice. A representation of the LRP variants proposed in this research work is presented in Figure 1. As mentioned before, these problems are LRPs and, therefore, different from the CLTPI.

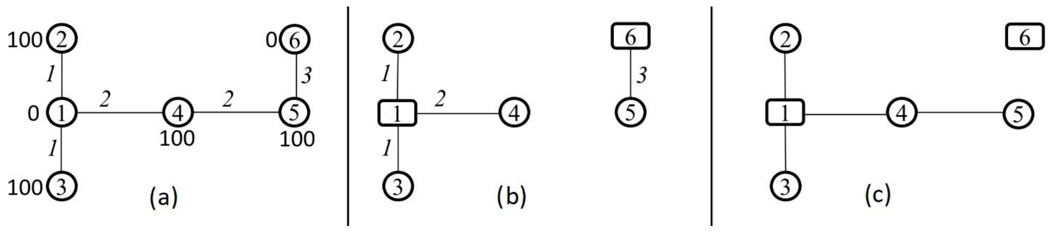

**Figure 1.** Araóz et al. proposed LRPs over tree-shaped graphs [11]. (**a**) Tree and edges. (**b**) Allocation to two facilities. (**c**) Allocation to one facility.

Another related problem is the Capacitated Facility Location/Network Problem (CFLNDP) [14], which arises from the CFLP. The CFLNDP considers a set of vertices representing clients with a given demand. A facility might be open at each client vertex and with a given capacity. Open facilities serve directly through a link to the clients assigned to the open facility. The authors consider an incomplete graph for the problem. The CFLNDP solutions are star-shaped networks originating from an open facility. In the CFLNDP, no capacity over the links is considered, and clients are served using direct links from the facilities. The main differences with the CLTPI are that there is no capacity over the links, there is no topological design of the network, all clients can open a facility, and the problem is defined over an incomplete graph.

Another two problems in the context of tree and location problems are the Capacitated Ring Tree problem (CRTP) [15] and the Ring-Tree Facility Location problem (RTFLP) [16,17]. Both problems consider a central depot to which the customers have to be connected by using a ring-tree-shaped network. Both problems consider two types of customers. Type 1 customers can be connected to the network by a tree or a ring, while type 2 customers need to be connected by rings. The rings are part of the inner level of the network, and trees are part of the outer level. The CRTP and the RTFLP consider a limit for the number of rings used to connect all customers and capacity constraints in the number of clients connected to each ring. The RTFLP considers facility opening costs (in the vertices that are part of the rings) and capacities in the outer level of the network, while the CRTP does not. The RTFLP can be considered a generalization of the CRTP. Among the most important differences between the CLTPI and RTFLP are that facilities can be opened for any customer (therefore becoming part of a ring) and that customers are connected to a central depot.

Finally, the Tree of Hubs Location Problem (THLP) is another problem related to the CLTPI. In this problem, a set of hubs must be located to transport people or commodities between the origin and destination points in the network. The main difference between the THLP and the classical Hub Location Problem (HLP), is that solutions in THLP must be tree-shaped. The THLP does not consider capacity over the hubs or the links between the hubs, differing from the CLTPI [18].

Table 1 presents a summary of problems related to the CLTPI. It presents the most relevant characteristics of each problem and some essential related works and methods used to solve them. The intention is to show some relevant related works, not an extensive literature review (for example, for the CFLP, the literature is quite comprehensive). The table shows that exact methods are predominant in the CFLNDP, the THLP, the RTFLP, and the LRP over tree-shaped networks. On the other hand, exact and heuristic methods are used for the CMST and the CFLP.

**Table 1.** Literature Review. Related Problems.

| **Problem** | **Characteristics** | **Authors** | **Solution Methods** |
|---|---|---|---|
| Capacitated minimum spanning tree | Tree-shaped network<br>Capacitated links<br>One source<br>No location | Chandy and Lo [7]<br>Ahuja et al. [19]<br>Ruiz et al. [20]<br>Ruiz [21] | Little's branch-and-bound<br>Multi-Exchange Neighborhood<br>BRKGA<br>Branch-and-cut algorithm |
| Capacitated facility location problem | Capacitated facilities<br>Facility location<br>Various sources<br>No network | Holmberg et al. [22]<br>Chudak and Williamson [23]<br>Rahmani and MirHassani [24] | Exact Algorithm<br>Local Search<br>Hybrid Firefly-GA |
| Location-routing problems over a tree network | Facility location<br>No capacities<br>Routing problem | Aráoz et al. [11] | Decomposition |
| Capacitated Facility Location/Network Problem | Facility location<br>Capacitated facilities<br>Star-shaped network<br>Various sources | Melkote and Daskin [14] | MILP |
| Ring-Tree Facility Location problem | Facility location<br>Ring-shaped network<br>One source | Abe et al. [16]<br>Abe et al. [17] | MILP<br>Branch-and-Price |
| Tree of Hubs Location Problem | Facility location<br>No capacities<br>Tree-shaped network | Contreras et al. [18]<br>Contreras et al. [25]<br>de Sá et al. [26] | Integer Programming<br>Decomposition<br>Benders decomposition |

## 3. Problem Description and Formulation

The Capacitated Location Tree Problem with Interconnection (CLTPI) focuses on optimizing the design of a distribution network to minimize the total cost. In this context, strategic decisions must be made regarding the placement of facilities, referred to as plants, to service a set of clients with specific demands. The network design must be robust. Hence, interconnections between facilities must be included to ensure service continuity, even in the event of a failure in any of them.

Interconnections, defined as links between two facilities, are exclusively utilized in failure situations and are not part of the network's regular operation. They aim to maintain the supply in case of a failure in any facility, ensuring the system's reliability. Additionally, the problem considers capacity constraints in both facilities and connections. Capacity constraints in facilities limit the amount of energy they can supply, while constraints on connections regulate energy flow between facilities and clients.

The formulation and modeling of the problem are carried out using graphs, allowing for an efficient representation of the relationships between facilities and connections. However, it is relevant to note that this problem is NP-Hard, signifying that its complexity increases significantly as the numerical instance size to be solved grows.

Despite an exhaustive literature review, no evidence of prior studies addressing the CLTPI has been found. It arises from the urgent need to design electrical networks for electric transportation systems capable of operating partially, even in the event of failures in power substations.

In summary, the primary objective of the problem is to minimize the total cost of network design, considering the cost of opening the required facilities, and designing the network to meet the required demands of all clients. This involves respecting capacity constraints in facilities and connections while strategically creating interconnections between facilities to maintain continuous service, at least partially, in the case of a failure in any facility.

The problem is formally defined with a complete directed graph, $G = (V, A)$, where $V$ is the set of vertices and $A = (i, j)|i \in V, j \in V|i \neq j$ is the set of arcs. The set $V$ contains two subsets of vertices: (i) $V_p \subset V$, the subset of plants, and (ii) $V_c \subset V$, the subset of clients. Therefore, $V = V_p \cup V_c$. For every arc $(i, j) \in A$, there is an associated cost, $c_{ij}$, that represents the cost of establishing a connection between vertices the $i \in V$ and $j \in V$. The number of available plants is given by $p = |V_p|$, the number of customers requiring the service by $c = |V_c|$, and the total number of vertices by $n = p + c = |V|$.

*Formulation*

A mixed-integer linear problem (MILP) flow-based formulation is proposed and presented in this section to model the CLTPI. This MILP formulation uses binary variables and continuous variables. Before formulating the problem, the following parameters are defined:

| | |
|---|---|
| $n$ | the total number of vertices in graph $G$, with $n = \vert V\vert$; |
| $p$ | the number of available locations for plants; |
| $cl$ | the number of customers to serve, with $cl = n - p$; |
| $d_j$ | the required demand of the customer $j \in V_c$; |
| $c_{ij}$ | the cost of establishing a connection using the arc $(i, j) \in A$; |
| $f_p$ | the opening cost for the plant (facility) $p \in V_p$; |
| $b_p$ | the limited capacity of the plant (facility) $p \in V_p$; |
| $Q$ | the capacity of the connection links. |

The required variables to formulate the problem are also defined:

$$y_p = \begin{cases} 1, & \text{if plant } p \text{ is open,} \quad p \in v_p \\ 0, & \text{otherwise.} \end{cases}$$

$$x_{ijp} = \begin{cases} 1, & \text{if arc } (i,j) \text{ is used in the network and is connected to plant } p \in V_p \\ 0, & \text{otherwise.} \end{cases}$$

$$t_{ijp} = \text{the flow passing through arc } (i,j) \text{ with origin in plant } p \in V_p$$

Using the previously defined parameters and variables, the following optimization formulation arises:

$$\text{Min} \quad \sum_{p \in V_p} f_p y_p + \sum_{p \in V_p} \sum_{i \in V} \sum_{j \in V_c} c_{ij} x_{ijp} \tag{1}$$

subject to

$$\sum_{p \in V_p} \sum_{\substack{i \in V \\ i \neq j}} x_{ijp} = 1 \qquad \forall j \in V_c \tag{2}$$

$$\sum_{p \in V_p} \sum_{\substack{i \in V_c \\ i \neq j}} d_i x_{jip} \leq Q - d_j \qquad \forall j \in V_c, \tag{3}$$

$$\sum_{i \in V} \sum_{\substack{j \in J_c \\ j \neq i}} d_j x_{ijp} \leq b_p y_p \qquad \forall p \in P \tag{4}$$

$$\sum_{p \in V_p} \sum_{\substack{i \in V \\ i \neq j}} t_{ijp} - \sum_{p \in V_p} \sum_{\substack{i \in V_c \\ i \neq j}} t_{jip} = d_j \qquad \forall j \in V_c \tag{5}$$

$$0 \leq t_{ijp} \leq Q \qquad \forall i \in V,\ j \in V_c,\ p \in P \tag{6}$$

$$x_{ijp} \in \{0, 1\} \qquad \forall i \in V, j \in V_c,\ p \in P \tag{7}$$

$$y_p \in \{0, 1\} \qquad \forall p \in P \tag{8}$$

The objective function (1) of the previously presented model minimizes the network's total construction cost. The first term represents the cost of opening the required facilities, while the second represents the cost of establishing the necessary connections for all the vertices in the network. On the other hand, the constraints (2) guarantee that every client vertex is connected to the network, while the constraints (3) limit the number of outgoing arcs from a client vertex, considering the capacity of the links. The plant's capacity is controlled with the constraints (4). To control the links' capacity and the connectivity of the solution, the flow constraints (5) and (6) are used.

This type of MILP model can be solved by using an exact method (EM) to obtain solutions for the CLTPI. For this research, the EM combines a dual-simplex algorithm with branch-and-bound techniques to solve CLTPI instances.

## 4. CLTPI Solution Algorithms

Instances of the CLTPI can be solved using exact and metaheuristic methods. Exact methods are based on mathematical programming formulations. On the other hand, metaheuristic methods are inspired by natural phenomena, physics phenomena, and animal behavior, among other sources. In this section, three solution approaches are presented. The first is an exact method (EM) based on the formulation presented in Section 3. The EM uses a branch-and-bound algorithm over the CLTPI formulation to obtain solutions. A good description of the branch-and-bound algorithm can be found in [27]. Since the CLTPI is an NP-Hard problem, solution methods like EMs are non-polynomial and consume many computational resources. In other words, the complexity of the EM grows exponentially as the size of the instances increases, and so does the computational effort.

The other two are metaheuristic evolutionary algorithms. It is known that large instances of NP-Hard problems are tough to solve using exact methods and require a significant computational effort to obtain solutions. Therefore, using metaheuristic algorithms that can provide good-quality solutions with a reasonable computational effort is relevant. Different kinds of heuristics and meta-heuristics have been used for solving combinatorial optimization problems like Genetic Algorithms [28], Simulated Annealing [29], Ant Colony Optimization [30], the Gravitational Search Algorithm [31], Tabu Search [32], and

Particle Swarm Optimization [33], among many others. To solve instances of the CLTPI, two metaheuristic optimization algorithms were selected: (i) the Particle Swarm Optimization algorithm (PSO) and (ii) the Biased Random-Key Genetic Algorithm (BRKGA). The reason for selecting both algorithms was their proven performance in solving tree-related combinatorial optimization problems (see [21,34]). In the following, a brief description of these metaheuristic optimization algorithms is provided.

### 4.1. Biased Random-Key Genetic Algorithm (BRKGA)

Bean [35] first proposed Random-Key Genetic Algorithms for solving combinatorial optimization problems. Combinatorial optimization problems, by their nature, are integer-based, meaning that variables must take integer values. The algorithm encompasses a population of random vectors that encode the solution to the stated problem. Later, Gonçalves and Resende [36] introduced the BRKGA, or Biased Random-Key Genetic Algorithm, for the first time. The BRKGA has been widely used in the context of combinatorial optimization problems like bin-packing problems [37], routing problems [38], clustering problems [39], regression testing [40], scheduling problems [41], and many other problems. Since the BRKGA has proven to be efficient in solving combinatorial optimization problems, frameworks and libraries have been developed, like the ones by Toso and Resende [42], Silva et al. [43], Andrade and Toso [44], and Oliveria et al. [45], among others.

The BRKGA has also been used in facility location and network design problems. In the context of facility location problems, it is essential to mention the works by Biajoli [46], Souto et al. [47], and Morais et al. [48]. In the context of tree problems, there are also relevant works like the ones by Ruiz et al. [20] and Pessoa et al. [49].

Random vectors are generally represented in the interval $[0,1]$ and are called chromosomes. Through the algorithm, a population evolves for several iterations (generations). In such algorithms, up to $L$ populations with $np$ individuals can be used, which evolve independently for a certain number of generations ($IG$). At this point, the best members of each population are introduced into the other populations to diversify them. This process is repeated every $IG$ generation.

On the other hand, bias in such algorithms is introduced by giving greater priority to parents with a better value in their objective function to pass on their genes to their descendants. The BRKGA starts by generating an initial population, $IP$, of random vectors, $v$, with values in the interval $[0,1]$. Then, every vector $v_i$ in the population $IP$ is subjected to a decodification procedure that returns a fitness value, $fv_i$. Using $fv_i$, the vectors $v_i$ are sorted in ascending (minimization function) or descending (maximization function) order. Once ordered, the first $p_e$ members are selected as the elite members of the population. The population $IP$ in the generation $g$ has $p_e$ elite members and $np - p_e$ non-elite members. To create the population $IP$ in the next generation, $g+1$, all elite members in the $g$ generation are copied without modification. Then, a small number of $p_m$ mutants is randomly generated and introduced into the population $IP$. The other $np - p_e - p_m$ individuals are created by using crossover. Crossover is performed by always using an elite member as a parent. The other parent could be either a non-elite or an elite member of the population. Elite parents have a greater priority, $0.5 > \rho_e < 1$, to pass on their genes to a descendant. Figure 2 shows a graphical description of how the next generation $g+1$ is created from the current generation $g$.

The BRKGA evolves until a stop criterion is met. The most common criterion is to stop after a given number of iterations without improvement is reached. In Figure 3, the flowchart of the BRKGA algorithm is shown.

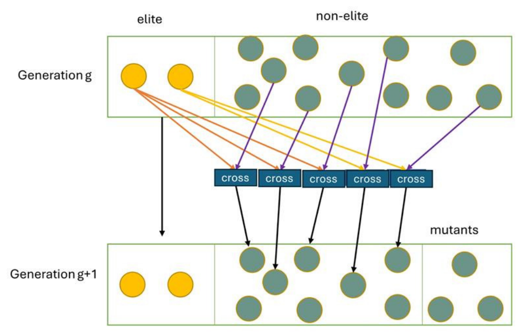

**Figure 2.** BRKGA crossover between elite and non-elite members in generation $g$ to produce generation $g + 1$.

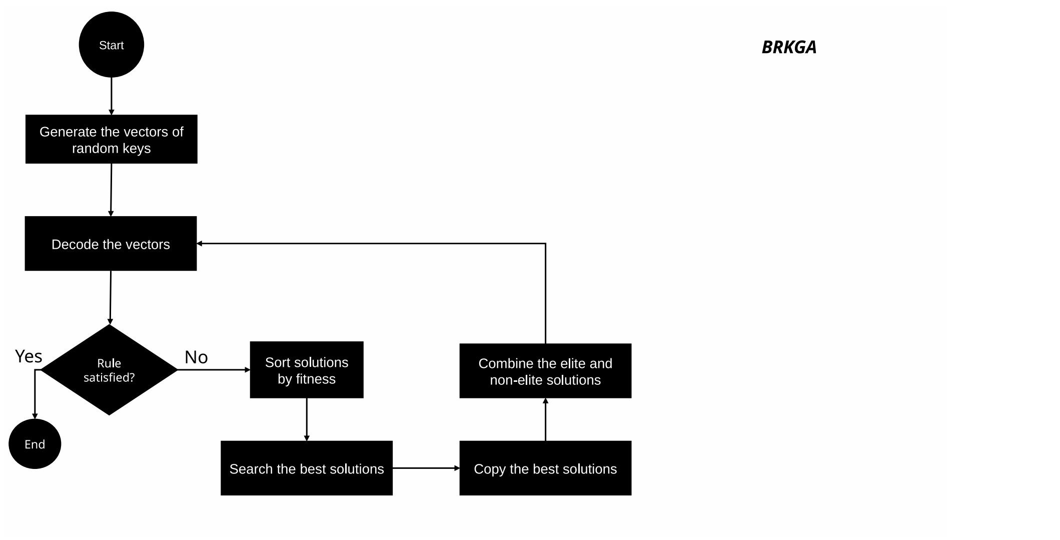
BRKGA
Start
Generate the vectors of random keys
Decode the vectors
Rule satisfied?
Yes
No
End
Sort solutions by fitness
Search the best solutions
Copy the best solutions
Combine the elite and non-elite solutions

**Figure 3.** BRKGA flow chart.

### 4.2. PSO

Particle Swarm Optimization (PSO) stands as an evolutionary computing technique within the domain of bioinspired systems. The PSO was first proposed by Kennedy and Eberhart [33]. The PSO is a very popular optimization algorithm that has been used in various topics and applications. In the context of combinatorial optimization problems, the PSO has been used in vehicle routing problems (VRPs) [50–52], facility location problems [53–55], location-routing problems [56,57], and scheduling problems [58,59], among many others. The PSO is a continuous optimization algorithm, although some discrete versions have been proposed, like the ones by [60,61]. Additionally, many variants of the PSO have been proposed. A good overview of such variants can be found in Jain et al. [62]. Another comprehensive review of the PSO can be found in [63].

Its inception involves the establishment of random populations as an initial solution. Subsequently, the algorithm evolves, generating a random velocity and potential solutions known as particles, traversing the problem space. These particles maintain a meticulous record of their coordinates within the problem space, intimately linked with the best solution, denoted as fitness. The imperative retention of the fitness value leads to its identification as *Pbest*. Furthermore, the best solution, derived from the global iteration of the PSO, is preserved. This pinnacle solution, irrespective of its origin, is identified as *Gbest* and represents the optimal state within the population. Central to the dynamic nature of PSO are the acceleration coefficients, denoted as $c1$ and $c2$ in the algorithmic formulation. These coefficients signify the weight of stochastic acceleration, influencing each particle's trajectory towards both its personal best *Pbest* and the global best *Gbest*. The manipulation of these acceleration constants introduces adjustments, thereby inducing tension within the system and steering the optimization process.

In this paper, a variant of regular PSO is used. The variant includes ideas from [64] for a variable inertia weight, from [65] for the variable acceleration coefficients $c1$ and $c2$, and the algorithm reset from [66]. Therefore, this variant uses variable instead of constant values for the parameters $c1$, $c2$, and $w$. As mentioned before, $c1$ and $c2$ are acceleration coefficients, while $w$ is the inertia coefficient. Also, the algorithm has a reset function that randomly initializes the particles except for one particle, which is initialized with the best solution the algorithm found before resetting. Such a function allows the algorithm to escape from local optima and is activated when a specific criterion is met (number of iterations without improvement). The reset function can be used several times, although experiments showed that such a function should be used one or two times at most.

The flow chart of the algorithm is shown in Figure 4.

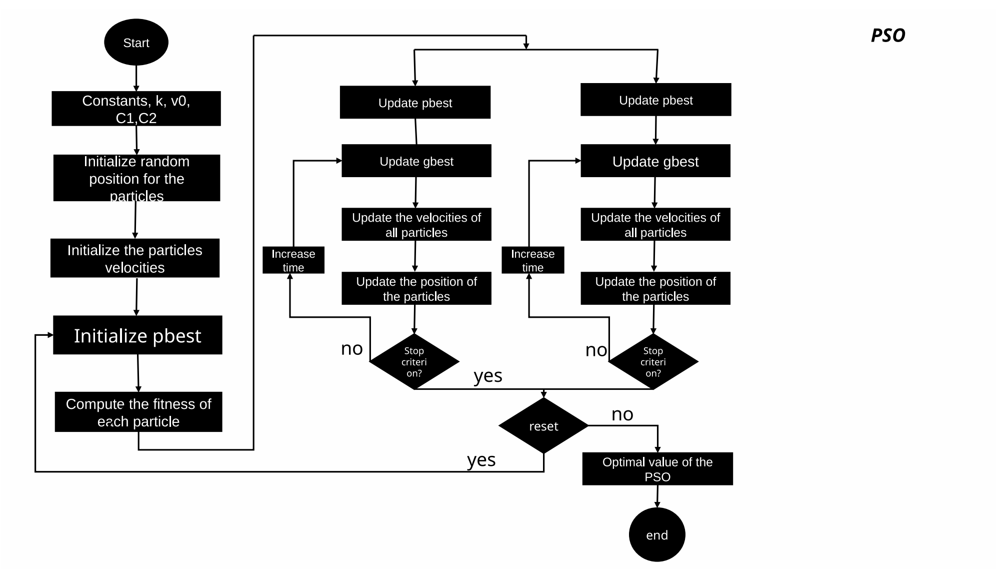
PSO
Start
Constants, k, v0, C1,C2
Initialize random position for the particles
Initialize the particles velocities
Initialize pbest
Compute the fitness of each particle
Update pbest
Update gbest
Update the velocities of all particles
Update the position of the particles
Increase time
Stop criterion?
no
yes
Update pbest
Update gbest
Update the velocities of all particles
Update the position of the particles
Increase time
Stop criterion?
no
reset
no
yes
Optimal value of the PSO
end

**Figure 4.** Parallel PSO with restart.

## 4.3. Decoder with Improvement Phase

The BRKGA and PSO are general-purpose optimization algorithms and require a decodification procedure to obtain solutions to a given problem. A cost-based decodification procedure is proposed and presented to obtain solutions for the CLTPI. This procedure is inspired by the one proposed in [21]. The decoder uses $n = |V|$ variables, equal to the total number of vertices in the graph $G$. Therefore, one variable is associated with each potential facility and each customer. The decoder receives, as input, $\mathcal{X}_p$, $\mathcal{X}_{cl}$, $G$, $n$, $p$, $cl$, $b$, $f$, $Q$, $d$, and $c$. Following a description of each input item is presented.

- $\mathcal{X}_p$ is the key vector associated with the set of potential facilities in the set $V_p$.
- $\mathcal{X}_{cl}$ is the key vector associated to the set of clients $V_c$.
- $n$ is the total number of vertices in the graph $G$; $n = |V|$.
- $p$ is the number of potential locations for facilities; $p = |V_p|$.
- $cl$ is the number of customers to serve; $cl = n - p = |V_c|$.
- $d$ is the demand vector for the customer $j \in V_c$.
- $c$ is the cost matrix for establishing a connection using the arc $(i, j) \in A$.
- $f$ is the opening cost for plants (facilities) in the set $V_p$.
- $b$ is the capacity limit for plant (facilities) in the set $V_p$.
- $Q$ is the maximum capacity of the connection links.
- A subtree is a rooted arborescence with an origin at a client vertex, $i \in V_{cl}$.

- An $s$-tree, $s$-$T_k$, is a particular case of a subtree with an origin at $k \in V_c$, where $k$ is directly connected ($x_{pkk} = 1$) to an opened plant vertex, $p \in V_p$.
- $MPR$ is the minimum number of facilities to open to cover the accumulated demand of all clients. MPR is computed using the variables $y_i$ defined in Section 3 and is solved using the following optimization problem:

$$\text{Min} \quad \sum_{p \in V_p} y_p \tag{9}$$

subject to

$$\sum_{p \in V_p} f_p y_p \geq \sum_{j \in V_c} d_j \tag{10}$$

$$y_p \in \{0, 1\} \quad \forall p \in P \tag{11}$$

The cost-based decoder uses two key vectors. The first key vector is related to the potential facilities and the second to the clients. The decoder receives as input two key vectors, $\mathcal{X}_p$ and $\mathcal{X}_c$, of the size $p$ and $cl$, respectively. Both key vectors are arranged in ascending order. The core idea behind the cost-based decoder is to scan the vertices in increasing order of the random keys in the vectors $\mathcal{X}_p$ and $\mathcal{X}_{cl}$ and try to assign the scanned vertex $i$ to its closest $s$-tree among the already-existing ones with enough available capacity. Here, the distance from the vertex $i$ to the $s$-tree $s$-$T_k$ is defined by the least $c_{ij}$ value for $j \in V(s\text{-}T_k)$.

As mentioned before, the cost-based decoder takes as input two random key vectors, $\mathcal{X}_p$ and $\mathcal{X}_{cl}$, the graph structure $G = (V, A)$, the potential plant set $V_p$, the client set $V_p$, the demand vector $d$, the capacity plant vector $f$, and the minimum required plants $MPR$ and returns an $n$-dimensional integer assignment vector, $a$. When $i \in V_c$, $a(i) = k$ indicates that the vertex $i$ is assigned to the $s$-tree $s$-$T_k$, and $a(i) = i$ indicates that the client $i$ is directly connected to an open facility. When $k \in V_p$, $a(k) = k$ indicates that the plant $k$ is open. The algorithm uses a vector, $s$, to keep the residual capacities of opened plants and $s$-trees to avoid allocating more demand than that supported by plants and links. Finally, the vector $r$ keeps a record of the plant to which an $s$-tree is directly connected.

Algorithm 1 gives the pseudo-code of the cost-based assignment assignment decoder. In lines 1 and 2, the assignment vector $a$, available capacity vector $s$, and plant assignment vector $r$ are initialized. The lists are initialized in lines 3 to 5. The assignation of clients is performed in the loop in lines 7 to 40. In each iteration, $i$ denotes the vertex to be assigned (initialized in line 6). In line 7, an ordered list is built using open plants and clients already assigned, which are stored in the list `ASSIGNED`. The loop in lines 10 to 29 assigns the vertex $i$ to the closest $s$-tree or plant with available capacity. If the client $i$ is assigned to an $s$-tree, the vectors $a$, $s$, and $r$ are updated in lines 13 to 16. If $i$ is directly connected to a plant, the vectors $a$, $s$, and $r$ are updated in lines 19 to 22. If assigned, the client $i$ is removed from the list `VERTICES` and included in the list `ASSIGNED` in lines 25 and 26. If there is no $s$-tree or plant with available capacity to accommodate $i$, then a new plant, $k$, is open, and the client $i$ is directly connected to it and included in the list `ASSIGNED` (lines 30 to 37).

The proposed decoder is a two-phase decoder. In the first phase, the key vector is decoded and transformed into a solution of the CLPTI, while in the second phase, the obtained solution is improved. For the CLTPI, the improvement phase re-optimizes $s$-trees by computing an MST since $s$-trees respect the link capacity constraints. The MST is computed for the set of vertices contained in the $s$-tree plus the plant to which the $s$-tree is connected using Prim's algorithm [6].

**Algorithm 1:** Pseudo-code for `cost-based assignment` decoder

```
procedure cost-based assignment
    Input: $\mathcal{X}_p, \mathcal{X}_c, V, A, V_p, V_c, f, Q, d, c, MPR$
    Output: Assignment array a
    $a(p) \leftarrow 0, s(p) \leftarrow f(p)$, for $p \in V_p$
    $a(i) \leftarrow 0, r(i) \leftarrow 0, s(i) \leftarrow Q$, for $i \in V_c$
    Initialize list VERTICES with vertices $1, \in V_c$ in increasing order of $\mathcal{X}_c$;
    Initialize list ASSIGNED with vertices $1, \ldots, MRP$ in increasing order of $\mathcal{X}_p$ ;
    Initialize list PLANTS with vertices $MPR, \ldots, p$ in increasing order of $\mathcal{X}_p$;
    $i \leftarrow FIRST$(VERTICES);
    while $i \neq$ nil do
        /* Try to connect $i$ to an existing subtree or plant */
        Sort vertices $j$ in ASSIGNED in increasing order of $c_{ij}$;
        $j \leftarrow FIRST$(ASSIGNED);
        while $j \neq$ nil and $a(i) == 0$ do
            $k \leftarrow a(j)$;
            if $k \in V_c$ and $s(k) \geq d_i$ and $s(r(k)) \geq d_i$ then
                $a(i) \leftarrow k$;
                $s(k) \leftarrow s(k) - d_i$;
                $r(i) \leftarrow k$;
                $s(r(k)) \leftarrow s(r(k)) - d_i$;
            end
            if $k \in V_p$ and $s(k) \geq d_i$ then
                $a(i) \leftarrow i$;
                $s(i) \leftarrow s(i) - d_i$;
                $r(i) \leftarrow k$;
                $s(k) \leftarrow s(k) - d_i$;
            end
            if $a(i) \neq 0$ then
                Remove $i$ from CANDIDATE list;
                Add $i$ to ASSIGNED list;
            end
            $j \leftarrow$ NEXT(ASSIGNED);
        end
        if $a(i) == 0$ then
            /* Open a new plant an connect vertex $i$ */
            $k \leftarrow FIRST$(PLANTS);
            $a(i) \leftarrow i$;
            $s(i) \leftarrow s(i) - d_i$;
            $s(k) \leftarrow s(k) - d_i$;
            Remove $i$ from CANDIDATE list;
            Add $i$ to ASSIGNED list;
            Remove $k$ from PLANTS list;
            Add $k$ to ASSIGNED list;
        end
        $i \leftarrow$ NEXT(VERTICES);
    end
    return
```

## 5. Computational Experiments

To test the performance of the proposed solution algorithms (EM, BRKGA, and PSO), 36 test instances from the literature were used. These test instances were proposed to the LRP and can be found at http://prodhonc.free.fr/Instances/instances_us.htm (accessed on 16 January 2025). These instances were chosen because their characteristics are applicable to the CLTPI. The size of the test instance ranged from 20 to 200 clients, and there were 5 to 10 potential facilities. The names of the test instances reflected the number of clients (first number) and the number of potential facilities (second number). The EM was developed in C++ Ubuntu 13.3.0 using GUROBI 9.1.1 as the solver. The BRKGA and PSO algorithms were also developed in C++ Ubuntu 13.3.0.

All the experiments were run on a PC with an AMD Ryzen 9 3950 processor at 3.5 GHz with 16 cores and 64 GB of RAM. Tables 2–4 show, respectively, the results of the exact algorithm (Formulation), BRKGA, and PSO using the benchmark test instances. The EM was run just once (the algorithm was deterministic), while the BRKGA and PSO were run five times for each test instance. For the EM, a maximum solution time of 72,000 s was defined. A comparison of the best solutions obtained by each solution algorithm and the average time required to find it is presented in Table 5.

For the BRKGA, the following parameters were used:

- $L = 50$, the number of populations used;
- $pe = 0.25$, the percentage of elite elements in each population;
- $pm = 0.10$, the percentage of mutants introduced in each population after crossover;
- $Th = 1.32$ the number of threads to be used (parallel computing);
- $ItEx = 40$, the number of iterations between exchanges of elite solutions among populations;
- $ElitetoEx = 1$, the number of elite solutions to exchange;
- $\rho = 0.65$, the probability that offspring inherits the vector component of its elite parent;
- $ItStop = 350$, the number of iterations without improvement in the BRKGA stopping criterion.

For the PSO, the following initial parameters were used:

- $L = 224$, the number of particles used;
- $c_1 = [2, 1.496]$, the acceleration coefficient towards the global best solution *Gbest*;
- $c_2 = [2, 1.496]$, the acceleration coefficient towards the particles' best solution *Pbest*;
- $w = [0.7928, 0.7578]$, the percentage of mutants introduced in each population after crossover;
- $ItStop = 350$, the number of iterations without improvement in the stopping criterion.

Note that the parameters $c1$, $c2$, and $w$ were modified every iteration when no new best solution was found.

In Table 2, column **Instance** gives the name of the test instance, while columns **UB** and **LB** show, respectively, the upper and lower bounds obtained by the EM. Column **Gap** shows the percentage deviation between the UB and LB, and **T16** shows the elapsed time used by the EM. For Tables 3 and 4, column **Instance** gives the name of the test instance, while columns **Best**, **Mean**, and **Worst** show, respectively, the best, the mean, and the worst value found by the corresponding algorithm after five runs. Columns **T1** and **T16** show the average time using one and sixteen CPU cores during five runs with the algorithms. Column $\boldsymbol{S(16)}$ shows the speedup of the parallel program using 16 CPU cores. The speedup was computed using

$$S(16) = \frac{\text{T1}}{\text{T16}}.$$

Finally, column $\sigma$ shows the objective function's standard deviation for each instance after five runs.

**Table 2.** Exact method's results.

| Instances | LB | UB | Gap | T16 |
|---|---:|---:|---:|---:|
| coord20-5-1.dat | 21,327.37 | 21,327.68 | 0.00 | 10.92 |
| coord20-5-1b.dat | 15,638.81 | 15,638.81 | 0.00 | 4.97 |
| coord20-5-2.dat | 22,893.29 | 22,893.29 | 0.00 | 5.08 |
| coord20-5-2b.dat | 14,050.44 | 14,051.02 | 0.00 | 3.15 |
| coord50-5-1.dat | 15,219.37 | 15,262.86 | 0.29 | 7200.64 |
| coord50-5-1b.dat | 15,065.81 | 15,098.14 | 0.21 | 7200.64 |
| coord50-5-2.dat | 29,563.77 | 29,582.63 | 0.06 | 7200.64 |
| coord50-5-2b.dat | 29,520.52 | 29,580.44 | 0.20 | 7200.64 |
| coord50-5-2BIS.dat | 16,845.06 | 16,869.25 | 0.14 | 7200.63 |
| coord50-5-2bBIS.dat | 17,836.50 | 17,879.67 | 0.24 | 7200.64 |
| coord50-5-3.dat | 11,099.39 | 11,135.09 | 0.25 | 7200.64 |
| coord50-5-3b.dat | 10,973.95 | 11,005.12 | 0.28 | 7200.64 |
| coord100-5-1.dat | 133,473.45 | 133,544.83 | 0.05 | 7200.72 |
| coord100-5-1b.dat | 133,255.56 | 133,320.51 | 0.05 | 7200.71 |
| coord100-5-2.dat | 97,292.59 | 97,352.20 | 0.06 | 7200.69 |
| coord100-5-2b.dat | 97,075.44 | 97,114.30 | 0.04 | 7200.70 |
| coord100-5-3.dat | 86,829.41 | 86,878.30 | 0.06 | 7200.71 |
| coord100-5-3b.dat | 86,630.17 | 86,683.47 | 0.06 | 7200.70 |
| coord100-10-1.dat | 155,463.85 | 155,564.86 | 0.06 | 7200.72 |
| coord100-10-1b.dat | 155,299.63 | 155,357.72 | 0.04 | 7200.72 |
| coord100-10-2.dat | 141,492.96 | 141,570.05 | 0.05 | 7200.74 |
| coord100-10-2b.dat | 141,354.23 | 141,416.36 | 0.04 | 7200.68 |
| coord100-10-3.dat | 136,552.56 | 139,911.96 | 2.46 | 7200.84 |
| coord100-10-3b.dat | 136,412.67 | 136,486.56 | 0.05 | 7200.84 |
| coord200-10-1.dat | 232,017.92 | 338,024.89 | 45.69 | 7212.86 |
| coord200-10-1b.dat | 231,458.68 | 318,817.21 | 37.74 | 7201.71 |
| coord200-10-2.dat | 261,216.53 | 339,855.34 | 30.10 | 7201.61 |
| coord200-10-2b.dat | 260,206.20 | 410,000.40 | 57.57 | 7202.89 |
| coord200-10-3.dat | 226,295.97 | 469,606.39 | 107.52 | 7201.38 |
| coord200-10-3b.dat | 225,896.31 | 497,752.47 | 120.35 | 7210.48 |
| coordGaspelle21-5.dat | 318.06 | 318.06 | 0.00 | 2.85 |
| coordGaspelle22-5.dat | 447.09 | 447.09 | 0.00 | 2.62 |
| coordGaspelle29-5.dat | 374.23 | 374.23 | 0.00 | 41.69 |
| coordGaspelle32-5.dat | 359.10 | 359.10 | 0.00 | 118.17 |
| coordGaspelle32-5b.dat | 383.41 | 383.41 | 0.00 | 529.44 |
| coordGaspelle36-5.dat | 399.37 | 399.37 | 0.00 | 14.89 |
| Average | | | 11.21 | 5221.63 |

The results in Table 2 show that the exact algorithm could find near-optimal solutions with gaps of less than 1% for instances with up to 100 clients except for the instance "coord100-10-3.dat". However, for instances with 200 clients, the results were poor, with gaps between 23.14% and 54.62%. This was expected since exact algorithms struggle with large test instances because the number of possible solutions grows exponentially. The average solution time for all test instances was 5221.63 s. For instances with 36 clients or fewer, the exact method found the optimal solution in less than 2 min except for the instance "coordGaspelle32-5b.dat", which required 529.44 s. For instances with 50 or more clients, the EM could not find the optimal solutions in 7200.00 s, although it found near-optimal quality solutions for instances with up to 100 clients. The EM sometimes surpassed the 7200.00 s limit due to the nature of the branch-and-bound procedure and the required time to retrieve the final LB and UB solutions.

**Table 3.** BRKGA with local search results.

| Instances | Best | Mean | Worst | T1 | T16 | $S(16)$ | $\sigma$ |
|---|---|---|---|---|---|---|---|
| coord20-5-1.dat | 21,410.21 | 21,410.21 | 21,410.21 | 1.43 | 0.44 | 3.26 | 0.00 |
| coord20-5-1b.dat | 15,734.84 | 15,734.84 | 15,734.84 | 1.37 | 0.67 | 2.05 | 0.00 |
| coord20-5-2.dat | 23,025.61 | 23,025.61 | 23,025.61 | 1.31 | 0.42 | 3.10 | 0.00 |
| coord20-5-2b.dat | 14,130.77 | 14,130.77 | 14,130.77 | 1.06 | 0.52 | 2.04 | 0.00 |
| coord50-5-1.dat | 15,474.99 | 15,495.23 | 15,513.15 | 10.13 | 1.43 | 7.07 | 14.77 |
| coord50-5-1b.dat | 15,291.78 | 15,293.18 | 15,295.27 | 6.67 | 0.83 | 8.05 | 1.91 |
| coord50-5-2.dat | 29,779.23 | 29,783.09 | 29,785.70 | 7.87 | 1.04 | 7.59 | 2.92 |
| coord50-5-2b.dat | 29,720.04 | 29,721.11 | 29,723.27 | 6.96 | 0.86 | 8.08 | 1.25 |
| coord50-5-2BIS.dat | 16,944.89 | 16,953.63 | 16,958.41 | 9.04 | 1.32 | 6.87 | 5.54 |
| coord50-5-2bBIS.dat | 17,949.03 | 17,950.48 | 17,952.82 | 6.97 | 0.88 | 7.88 | 1.75 |
| coord50-5-3.dat | 11,165.78 | 11,172.11 | 11,177.01 | 8.92 | 1.25 | 7.15 | 4.92 |
| coord50-5-3b.dat | 11,021.83 | 11,023.84 | 11,026.24 | 8.56 | 1.14 | 7.54 | 1.84 |
| coord100-5-1.dat | 133,855.86 | 133,875.58 | 133,896.73 | 38.92 | 4.19 | 9.29 | 15.43 |
| coord100-5-1b.dat | 133,568.12 | 133,574.26 | 133,587.49 | 36.58 | 3.68 | 9.94 | 7.66 |
| coord100-5-2.dat | 97,457.21 | 97,468.77 | 97,480.46 | 38.44 | 4.17 | 9.23 | 10.94 |
| coord100-5-2b.dat | 97,171.49 | 97,183.15 | 97,196.07 | 47.43 | 4.74 | 10.01 | 9.79 |
| coord100-5-3.dat | 87,090.96 | 87,112.36 | 87,126.43 | 38.57 | 4.01 | 9.62 | 15.75 |
| coord100-5-3b.dat | 86,825.72 | 86,839.26 | 86,845.09 | 38.91 | 4.04 | 9.62 | 7.84 |
| coord100-10-1.dat | 155,854.58 | 155,887.32 | 155,942.04 | 37.97 | 3.89 | 9.75 | 33.51 |
| coord100-10-1b.dat | 155,601.83 | 155,620.79 | 155,641.85 | 42.13 | 3.96 | 10.65 | 14.39 |
| coord100-10-2.dat | 142,170.27 | 142,201.14 | 142,223.77 | 40.23 | 4.08 | 9.85 | 21.26 |
| coord100-10-2b.dat | 142,005.11 | 142,016.26 | 142,027.49 | 39.94 | 3.81 | 10.48 | 10.05 |
| coord100-10-3.dat | 137,071.86 | 137,088.64 | 137,113.42 | 46.38 | 4.92 | 9.43 | 18.83 |
| coord100-10-3b.dat | 136,843.91 | 136,869.14 | 136,887.26 | 39.42 | 4.23 | 9.32 | 18.68 |
| coord200-10-1.dat | 238,467.66 | 238,514.61 | 238,570.86 | 190.47 | 17.78 | 10.71 | 38.00 |
| coord200-10-1b.dat | 237,804.70 | 237,834.71 | 237,858.06 | 191.86 | 17.67 | 10.86 | 27.15 |
| coord200-10-2.dat | 277,684.21 | 277,713.01 | 277,739.12 | 135.04 | 12.44 | 10.85 | 19.50 |
| coord200-10-2b.dat | 277,284.72 | 277,298.56 | 277,320.34 | 266.08 | 19.38 | 13.73 | 14.02 |
| coord200-10-3.dat | 236,137.19 | 236,182.69 | 236,271.29 | 150.35 | 15.02 | 10.01 | 52.61 |
| coord200-10-3b.dat | 235,591.97 | 235,614.08 | 235,635.44 | 172.30 | 15.47 | 11.14 | 17.26 |
| coordGaspelle21-5.dat | 368.49 | 368.49 | 368.49 | 1.58 | 0.31 | 5.05 | 0.00 |
| coordGaspelle22-5.dat | 491.39 | 491.39 | 491.39 | 1.53 | 0.29 | 5.27 | 0.00 |
| coordGaspelle29-5.dat | 395.37 | 395.37 | 395.37 | 1.99 | 0.34 | 5.94 | 0.00 |
| coordGaspelle32-5.dat | 384.76 | 384.76 | 384.76 | 3.48 | 0.51 | 6.85 | 0.00 |
| coordGaspelle32-5b.dat | 410.68 | 413.25 | 415.90 | 3.19 | 0.49 | 6.57 | 2.61 |
| coordGaspelle36-5.dat | 400.06 | 400.06 | 400.06 | 2.63 | 0.39 | 6.77 | 0.00 |
| Average | | | | 46.55 | 4.46 | 8.10 | 10.84 |

The results for the BRKGA with a local search are presented in Table 3. The algorithm was run using one and sixteen CPU cores, and the average running times are shown in columns *T1* and *T16*, respectively. A predefined group of seeds was used for each of the five times the BRKGA was executed. Therefore, the result with one or sixteen CPU cores was identical for the same seed. The results show that using multiple threads significantly reduced the average elapsed time to solve the CLTPI instances. On average, using sixteen CPU cores was 8.10 times faster than using one CPU core. The best reduction in the running time achieved was 19.38, for instance, by "coord200-10-2b.dat", where the BRKGA was 13.73 times faster when using sixteen CPU cores instead of one CPU core. The maximum average running time using 16 CPU cores was 19.38, for instance, with "coord200-10-2b.dat". Therefore, it can be assumed that the computational effort used by the BRKGA was reasonably low. On the other hand, the BRKGA showed good precision for all instances, with an average variance of 10.84. The quality of the solutions, considering the best solution found by the BRKGA, will be discussed after presenting Table 5.

**Table 4.** PSO with local search results.

| Instances | Best | Mean | Worst | T1 | T16 | $S(16)$ | $\sigma$ |
|---|---:|---:|---:|---:|---:|---:|---:|
| coord20-5-1.dat | 21,410.21 | 21,410.21 | 21,410.21 | 1.94 | 0.32 | 6.04 | 0.00 |
| coord20-5-1b.dat | 15,734.84 | 15,734.84 | 15,734.84 | 1.94 | 0.34 | 5.78 | 0.00 |
| coord20-5-2.dat | 23,025.61 | 23,025.61 | 23,025.61 | 2.08 | 0.31 | 6.66 | 0.00 |
| coord20-5-2b.dat | 14,130.77 | 14,130.77 | 14,130.77 | 1.85 | 0.30 | 6.08 | 0.00 |
| coord50-5-1.dat | 15,464.31 | 15,471.71 | 15,480.60 | 8.87 | 1.09 | 8.17 | 5.42 |
| coord50-5-1b.dat | 15,291.78 | 15,294.44 | 15,297.11 | 7.62 | 0.87 | 8.76 | 4.92 |
| coord50-5-2.dat | 29,777.33 | 29,779.52 | 29,781.82 | 10.55 | 1.13 | 9.32 | 5.93 |
| coord50-5-2b.dat | 29,720.13 | 29,720.48 | 29,720.58 | 7.00 | 0.80 | 8.69 | 3.85 |
| coord50-5-2BIs.dat | 16,937.74 | 16,943.61 | 16,948.93 | 7.76 | 0.95 | 8.15 | 4.74 |
| coord50-5-2bBIS.dat | 17,949.42 | 17,949.94 | 17,950.50 | 6.79 | 0.83 | 8.22 | 4.39 |
| coord50-5-3.dat | 11,157.09 | 11,166.28 | 11,172.70 | 9.34 | 1.13 | 8.30 | 5.20 |
| coord50-5-3b.dat | 11,021.20 | 11,023.84 | 11,024.77 | 7.76 | 0.93 | 8.31 | 4.45 |
| coord100-5-1.dat | 133,846.71 | 133,853.25 | 133,863.27 | 32.04 | 3.15 | 10.17 | 12.84 |
| coord100-5-1b.dat | 133,560.10 | 133,570.50 | 133,582.87 | 36.67 | 3.44 | 10.66 | 15.34 |
| coord100-5-2.dat | 97,429.58 | 97,450.19 | 97,477.21 | 26.69 | 2.78 | 9.60 | 11.77 |
| coord100-5-2b.dat | 97,171.09 | 97,175.50 | 97,178.91 | 34.88 | 3.75 | 9.31 | 15.10 |
| coord100-5-3.dat | 87,054.52 | 87,075.81 | 87,088.46 | 32.30 | 3.21 | 10.07 | 13.88 |
| coord100-5-3b.dat | 86,826.46 | 86,830.91 | 86,834.22 | 37.60 | 3.55 | 10.59 | 14.35 |
| coord100-10-1.dat | 155,835.28 | 155,860.79 | 155,886.41 | 40.04 | 5.46 | 7.33 | 14.64 |
| coord100-10-1b.dat | 155,575.83 | 155,603.82 | 155,624.14 | 33.39 | 4.42 | 7.55 | 18.69 |
| coord100-10-2.dat | 142,178.31 | 142,194.02 | 142,212.03 | 36.10 | 5.29 | 6.83 | 11.61 |
| coord100-10-2b.dat | 141,986.55 | 141,996.55 | 142,005.45 | 42.74 | 5.95 | 7.18 | 14.45 |
| coord100-10-3.dat | 137,060.42 | 137,078.48 | 137,094.14 | 26.42 | 3.83 | 6.91 | 18.30 |
| coord100-10-3b.dat | 136,837.49 | 136,857.38 | 136,866.61 | 30.72 | 4.15 | 7.39 | 18.89 |
| coord200-10-1.dat | 238,421.87 | 238,459.07 | 238,485.21 | 70.61 | 8.57 | 8.24 | 28.77 |
| coord200-10-1b.dat | 237,774.64 | 237,805.68 | 237,853.09 | 65.45 | 7.91 | 8.27 | 40.84 |
| coord200-10-2.dat | 277,667.09 | 277,681.84 | 277,701.03 | 100.86 | 12.09 | 8.34 | 33.35 |
| coord200-10-2b.dat | 277,229.42 | 277,248.71 | 277,261.81 | 103.84 | 12.20 | 8.51 | 43.35 |
| coord200-10-3.dat | 236,096.69 | 236,136.96 | 236,176.00 | 58.26 | 7.05 | 8.27 | 34.31 |
| coord200-10-3b.dat | 235,573.04 | 235,593.38 | 235,615.59 | 70.55 | 8.45 | 8.35 | 34.46 |
| coordGaspelle21-5.dat | 364.63 | 367.49 | 368.49 | 2.03 | 0.32 | 6.40 | 1.44 |
| coordGaspelle22-5.dat | 491.39 | 491.39 | 491.39 | 1.90 | 0.32 | 5.91 | 0.00 |
| coordGaspelle29-5.dat | 395.37 | 395.37 | 395.37 | 2.92 | 0.42 | 6.86 | 0.00 |
| coordGaspelle32-5.dat | 384.76 | 384.76 | 384.76 | 3.40 | 0.48 | 7.01 | 0.00 |
| coordGaspelle32-5b.dat | 410.68 | 414.12 | 417.63 | 3.46 | 0.50 | 6.97 | 2.19 |
| coordGaspelle36-5.dat | 400.06 | 400.06 | 400.06 | 4.11 | 0.56 | 7.39 | 0.00 |
| Average | | | | 26.96 | 3.25 | 7.96 | 12.15 |

The PSO computational results are presented in Table 3. As with the BRKGA, the PSO was run using one and sixteen CPU cores, and the average running times are shown in columns *T1* and *T16*, respectively. As with the BRKGA, the PSO results with one or sixteen CPU cores were identical for each instance and the same seed. Again, the results show that using multiple threads significantly reduced the average elapsed time needed to solve the CLTPI instances. For the PSO, on average, using sixteen CPU cores was 7.96 times faster than using one CPU core. The best reduction in the running time was achieved, for instance, by "coord100-5-1b.dat", where the PSO was 10.66 times faster when using sixteen CPU cores instead of one CPU core. The maximum average running time using 16 CPU cores was 12.20, for instance, with "coord200-10-2b.dat". Therefore, it can be assumed that the computational effort used by the PSO was lower than the BRKGA. On the other hand, the PSO also showed good precision for all instances, with an average variance of 12.15. The quality of the solutions, considering the best solution found by the PSO, will be discussed later in this section.

For the BRKGA and PSO, the speedup results are presented graphically in Figures 5 and 6. The BRKGA had a better speedup (8.10) than the PSO (7.96) when using 16 CPU cores. However, the PSO still had shorter solution times than the BRKGA. The reason was that iterations in the PSO were faster than in the BRKGA. The BRKGA took more time in each iteration due to the crossover operation. The efficiency of parallelization was around 50% for both algorithms. This efficiency was low due to the repeatability feature included in both algorithms. This means the algorithms had to return the same solution for a single given seed. To achieve the repeatability feature, the random numbers used in each iteration (in both algorithms) were generated using a single core, reducing the overall parallelization efficiency. The repeatability feature was lost if the generation of random numbers in each iteration was parallelized.

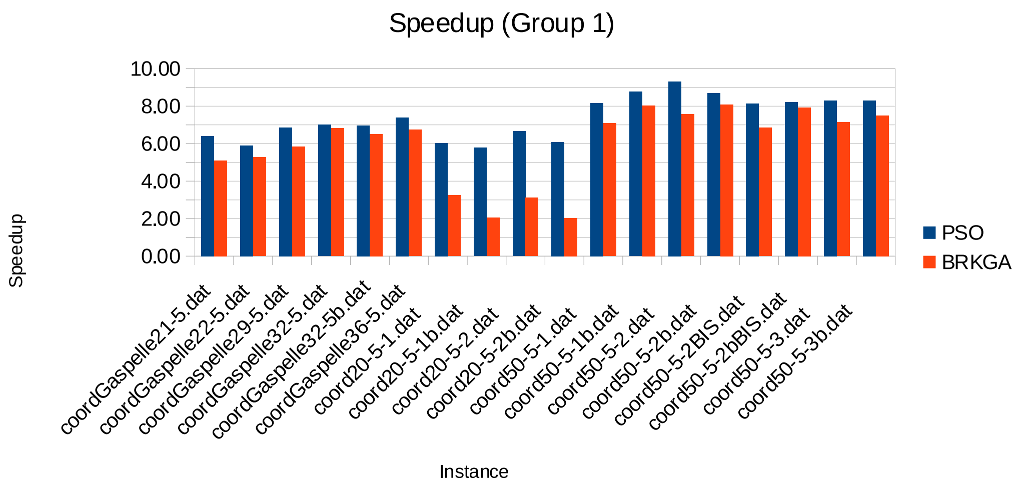

**Figure 5.** Speedup for instances with at most 50 vertices.

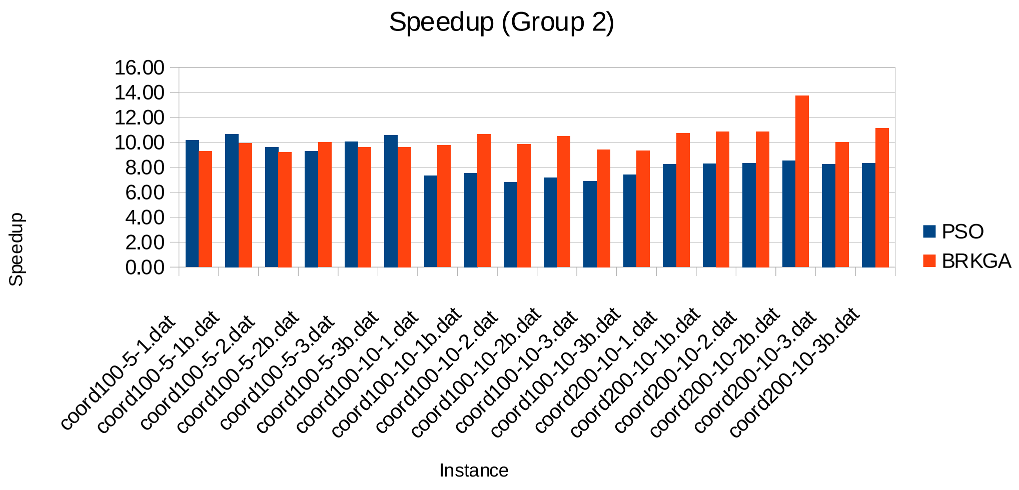

**Figure 6.** Speedup for instances with 100 vertices or more.

Table 5 compares the results of the three proposed solution algorithms. Column **BKS** presents the best-known solution for each instance, while columns **Gap** present the percentage of variation concerning the best solution found by each algorithm with the LB found by the EM. Values in bold in these columns indicate that the corresponding algorithm found the best-known solution. Finally, **T16** shows the elapsed time when using 16 CPU cores with the EM, PSO, and BRKGA solution algorithms. Regarding the best-known solutions, the EM algorithm reached the best-known solutions for 29 of the 36 benchmark instances, while the PSO reached the best-known solution for 7 of the 36 benchmark instances. The BRKGA did not reach any best-known solution. The results show that the EM algorithm found the best solutions for all instances with up to 100 clients (see Figures 7 and 8), except for "coord100-10-3b.dat," where the PSO obtained the best-known solution. The PSO obtained all the best-known solutions in the group of instances with

200 clients. The EM obtained poor results for instances with 200 clients with gaps between 30.10% and 120.35%, which were huge. The EM required much more computational effort than the BRKGA and PSO. Its average time was 5221.63 s, far greater than the 4.46 and 3.25 s used by the BRKA and PSO, respectively. The PSO obtained the lowest average gap with 1.50, the BRKGA's was 1.54, and the EM's was 9.96. The PSO and BRKGA found better solutions for large instances than the EM because the complexity of the PSO and BRKGA did not grow exponentially as it did for the EM.

**Table 5.** Comparison of best solutions between algorithms.

| **Instances** | **BKS** | **EM** | | **PSO** | | **BRKGA** | |
|---|---:|---:|---:|---:|---:|---:|---:|
| | | **gap** | **T16** | **gap** | **T16** | **Gap** | **T16** |
| coord20-5-1.dat | 21,327.68 | **0.00** | 10.92 | 0.39 | 0.32 | 0.39 | 0.44 |
| coord20-5-1b.dat | 15,638.81 | **0.00** | 4.97 | 0.61 | 0.34 | 0.61 | 0.67 |
| coord20-5-2.dat | 22,893.29 | **0.00** | 5.08 | 0.58 | 0.31 | 0.58 | 0.42 |
| coord20-5-2b.dat | 14,051.02 | **0.00** | 3.15 | 0.57 | 0.30 | 0.57 | 0.52 |
| coord50-5-1.dat | 15,262.86 | **0.29** | 7200.64 | 1.61 | 1.09 | 1.68 | 1.43 |
| coord50-5-1b.dat | 15,098.14 | **0.21** | 7200.64 | 1.50 | 0.87 | 1.50 | 0.83 |
| coord50-5-2.dat | 29,582.63 | **0.06** | 7200.64 | 0.72 | 1.13 | 0.73 | 1.04 |
| coord50-5-2b.dat | 29,580.44 | **0.20** | 7200.64 | 0.68 | 0.80 | 0.68 | 0.86 |
| coord50-5-2BIs.dat | 16,869.25 | **0.14** | 7200.63 | 0.55 | 0.95 | 0.59 | 1.32 |
| coord50-5-2bBIS.dat | 17,879.67 | **0.24** | 7200.64 | 0.63 | 0.83 | 0.63 | 0.88 |
| coord50-5-3.dat | 11,135.09 | **0.25** | 7200.64 | 0.44 | 1.13 | 0.52 | 1.25 |
| coord50-5-3b.dat | 11,005.12 | **0.28** | 7200.64 | 0.43 | 0.93 | 0.44 | 1.14 |
| coord100-5-1.dat | 133,544.83 | **0.05** | 7200.72 | 0.28 | 3.15 | 0.29 | 4.19 |
| coord100-5-1b.dat | 133,320.51 | **0.05** | 7200.71 | 0.23 | 3.44 | 0.23 | 3.68 |
| coord100-5-2.dat | 97,352.20 | **0.06** | 7200.69 | 0.14 | 2.78 | 0.17 | 4.17 |
| coord100-5-2b.dat | 97,114.30 | **0.04** | 7200.70 | 0.10 | 3.75 | 0.10 | 4.74 |
| coord100-5-3.dat | 86,878.30 | **0.06** | 7200.71 | 0.26 | 3.21 | 0.30 | 4.01 |
| coord100-5-3b.dat | 86,683.47 | **0.06** | 7200.70 | 0.23 | 3.55 | 0.23 | 4.04 |
| coord100-10-1.dat | 155,564.86 | **0.06** | 7200.72 | 0.24 | 5.46 | 0.25 | 3.89 |
| coord100-10-1b.dat | 155,357.72 | **0.04** | 7200.72 | 0.18 | 4.42 | 0.19 | 3.96 |
| coord100-10-2.dat | 141,570.05 | **0.05** | 7200.74 | 0.48 | 5.29 | 0.48 | 4.08 |
| coord100-10-2b.dat | 141,416.36 | **0.04** | 7200.68 | 0.45 | 5.95 | 0.46 | 3.81 |
| coord100-10-3.dat | 137,060.42 | 2.46 | 7200.84 | **0.37** | 3.83 | 0.38 | 4.92 |
| coord100-10-3b.dat | 136,486.56 | **0.05** | 7200.84 | 0.31 | 4.15 | 0.32 | 4.23 |
| coord200-10-1.dat | 238,421.87 | 45.69 | 7212.86 | **2.76** | 8.57 | 2.78 | 17.78 |
| coord200-10-1b.dat | 237,774.64 | 37.74 | 7201.71 | **2.73** | 7.91 | 2.74 | 17.67 |
| coord200-10-2.dat | 277,667.09 | 30.10 | 7201.61 | **6.30** | 12.09 | 6.30 | 12.44 |
| coord200-10-2b.dat | 277,229.42 | 57.57 | 7202.89 | **6.54** | 12.20 | 6.56 | 19.38 |
| coord200-10-3.dat | 236,096.69 | 107.52 | 7201.38 | **4.33** | 7.05 | 4.35 | 15.02 |
| coord200-10-3b.dat | 235,573.04 | 120.35 | 7210.48 | **4.28** | 8.45 | 4.29 | 15.47 |
| coordGaspelle21-5.dat | 318.06 | **0.00** | 2.85 | 14.64 | 0.32 | 15.86 | 0.31 |
| coordGaspelle22-5.dat | 447.09 | **0.00** | 2.62 | 9.91 | 0.32 | 9.91 | 0.29 |
| coordGaspelle29-5.dat | 374.23 | **0.00** | 41.69 | 5.65 | 0.42 | 5.65 | 0.34 |
| coordGaspelle32-5.dat | 359.10 | **0.00** | 118.17 | 7.14 | 0.48 | 7.14 | 0.51 |
| coordGaspelle32-5b.dat | 383.41 | **0.00** | 529.44 | 7.11 | 0.50 | 7.11 | 0.49 |
| coordGaspelle36-5.dat | 399.37 | **0.00** | 14.89 | 0.17 | 0.56 | 0.17 | 0.39 |
| Average | | 9.96 | 5221.63 | 1.50 | 3.25 | 2.37 | 4.46 |

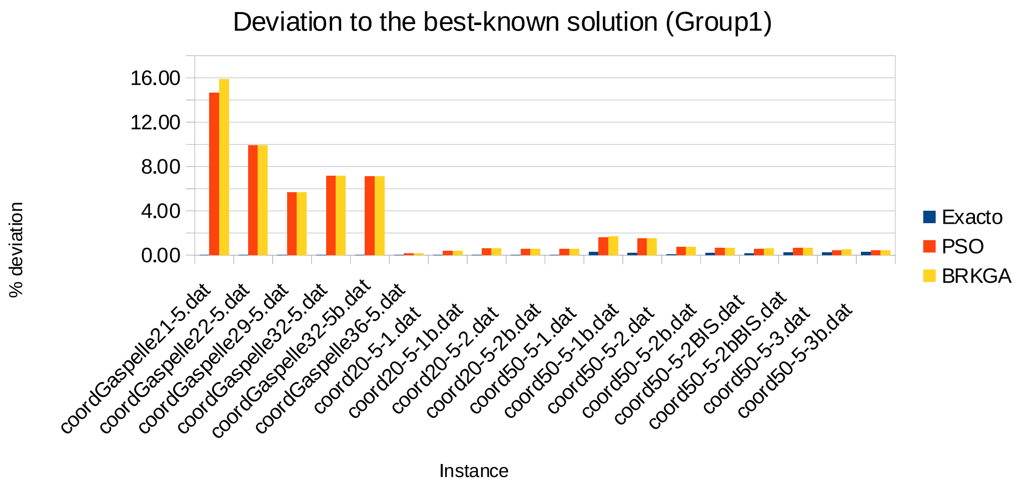

**Figure 7.** Deviation to the best LB for instances with at most 50 vertices.

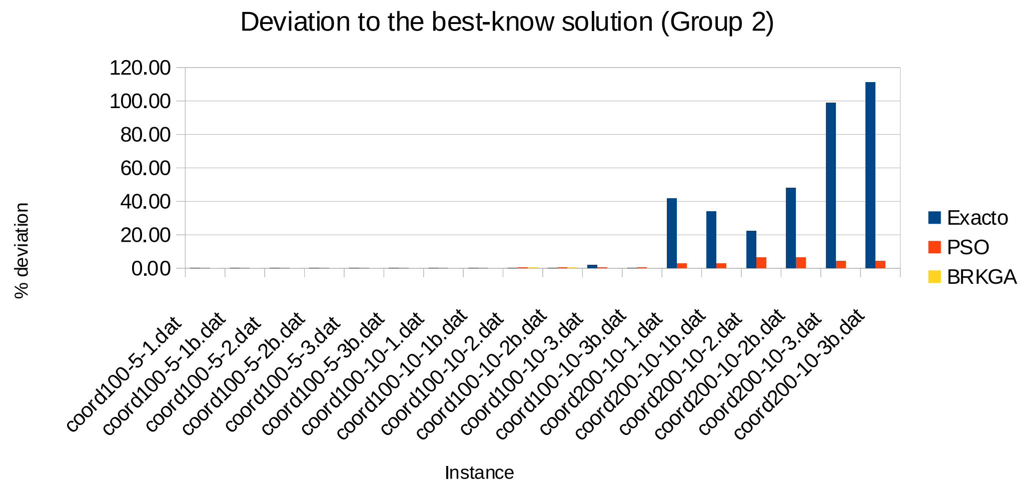

**Figure 8.** Deviation to the best LB for instances with 100 vertices or more.

## 6. Case Study

The CLTPI was inspired by the need to redesign an electrical network for an electric public transportation system. The goal was to obtain a robust network capable of partial functioning in case of substantial energy disruptions due to different causes. The network requires connecting traction substations (clients) with the high-voltage electrical substations (plants). The electricity transportation system serves more than 1,057,461,875 passengers per year. The service is available from 5:00 a.m. to 11:59 p.m. during work days. On Saturday, the service starts at 7:00 a.m. and ends at 11:59 p.m. Finally, on Sundays and holidays, the service is available from 7:00 a.m. to 11:59 p.m. Failure to provide the service would strand thousands of people per hour, resulting in work delays and a decline in the local economy.

The network extends over 226.49 km, distributed around the city. Within these kilometers, there are 174 traction substations. A traction substation converts electrical energy from the general network into the appropriate voltage, current, and frequency so that it can be used by the vehicles on the network. The power required by the traction substations is provided by high-voltage electrical substations. In the context of the CLTPI, the traction substations represent the clients, while the high-voltage electrical substations represent the plants.

The problem of energy shortages in the transportation system lies in the growth of the population using this kind of transportation to travel around the city. Their growth means that track substations require more energy consumption. Energy consumption is increasing, causing the high-voltage electrical substations to be unable at some point to supply track substations' demand. This consumption by track substations and the capacity of the high-voltage electrical substations being unable to provide what is requested leads to a redesign of connections according to the capacities of each high-voltage electrical substation and track substation's demand. Track substation demand considers the peak hours, allowing the network operator to know the maximum demand that produces the highest energy consumption. The excess consumption produces energy shortages and, in some cases, the shutdown of the transportation system, causing travel delays. After a shutdown, the system needs to be restarted, leading to an excess demand for current, which could heat the connections and cause power shortages again.

Nine electrical substations were considered for the case study. Three are already functioning (plants 0, 1, and 2), so their opening cost is zero. The other six have a specific opening cost and a given capacity. As for the track substations, 174 were considered. As previously mentioned, the idea was to redesign the network to obtain a robust one capable of partial functioning in case of critical energy disruptions.

The strategy consisted of selecting high-voltage electrical substations that provide enough energy to provide the required power to all the track substations; it was not necessary to open all the high-voltage electrical substations, but all the track substations needed to receive the required energy from the selected high-voltage electrical substations. It was essential to maintain the system's stability while operating the electrical network, trying not to saturate the high-voltage electrical substations to prevent power outages or shortages.

Table 6 presents the details for the high-voltage electrical substations. As previously mentioned, plants 0, 1, and 2 are operational and have zero opening costs. The capacities are expressed in MegaVolts-Ampers (MVA), and the opening costs are expressed in money units. The opening cost for each potential facility considers building and grid connection costs and other associated costs.

**Table 6.** Capacities and opening for high-voltage electrical substations.

| Plant Location | Capacity | Opening Cost |
|---|---:|---:|
| Facility 0 | 240 | USD 0 |
| Facility 1 | 120 | USD 0 |
| Facility 2 | 120 | USD 0 |
| Facility 3 | 240 | USD 4500 |
| Facility 4 | 120 | USD 2000 |
| Facility 5 | 300 | USD 5625 |
| Facility 6 | 300 | USD 5625 |
| Facility 7 | 300 | USD 5625 |
| Facility 8 | 300 | USD 5625 |

For every track substation, a demand of 4.4 MVA was considered. The location of track substations over Mexico City and its Metropolitan area are shown in Figure 9.

*Case Study Results*

An instance file was created using the information presented in the previous section to find a solution proposal for the case study. The instance was solved using the EM, the BRKGA, and PSO. As expected, the EM algorithm could not find a solution since it struggled with instances with more than 100 clients (track substations). On the other hand, the PSO and BRKGA obtained very similar solutions. Both solutions showed that potential facility (high-voltage substation) 8 had to be opened. The difference between the PSO and BRKGA solutions was the way some of the traction substations were connected. A summary of the results is presented in Table 7.

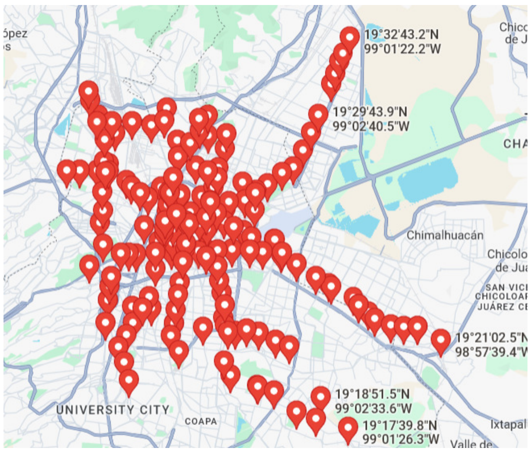

**Figure 9.** Location of track substations for the electric public transportation system.

**Table 7.** Case study results using EM, BRKGA and PSO.

| **Algorithm** | **Solution Cost** | **Time (s)** |
|---|---:|---:|
| EM | No solution found | 7200.00 |
| BRKGA | USD 5629.11 | 53.98 |
| PSO | USD 5629.13 | 94.62 |

## 7. Conclusions and Future Research

In this research paper, the Location Tree Problem with Interconnections was introduced. The problem was inspired by the need to design robust networks for electric transportation systems. This new combinatorial optimization problem is of the type NP-Hard, and a valid MILP formulation was proposed. An exact solution method (EM) based on the formulation was developed and used to solve benchmark instances of the problem. The EM obtained excellent results for small- and medium-size test instances. For larger instances, two heuristics (PSO, BRKGA) were also proposed and developed to solve instances of the problem. The two metaheuristic algorithms used a cost-based decoder to obtain solutions for the problem. The cost-based decoder always led to feasible solutions, avoiding the need to implement a feasibility recovery procedure. One of the metaheuristic algorithms was based on the BRKGA, while the second was based on a PSO algorithm. Computational experiments were performed using the three solution approaches to solve 36 benchmark instances of the problem. The results of the computational experiments show that the EM solution approach obtained the best solutions for the benchmark instances with 100 or fewer clients. In comparison, the PSO algorithm obtained the best solutions for the benchmark instances with 200 clients. It is important to note that the computational effort required by the EM algorithm was much larger than that of the PSO and BRKGA. Considering the best-known solutions, the EM found 29 of them, while the PSO found the other seven.

Additionally, a case study regarding an existing electrical transportation system was presented. The objective of the case study was to redesign the network so that it could function partially in the event of critical failures in the power supply. For the case study, the results show that a new electrical substation has to be opened to allow the grid to partially maintain service in the event of critical failures in the power supply.

Finally, future research can include the use of hybrid-solution approaches. For example, a metaheuristic (PSO, BRKGA, SA, ACO, GSA, etc.) with Tabu Search could be used. Other solution decoders and more complex local search neighborhoods should also be tried. Such hybrid approaches could lead to better solutions with similar or less computational effort.

**Author Contributions:** Conceptualization, E.R.-y.-R. and N.M.-A.; methodology, N.M.-A. and E.R.-y.-R.; programming, N.M.-A.; validation, E.R.-y.-R., N.M.-A. and S.R.-R.; formal analysis, N.M.-A.; investigation, N.M.-A.; resources, E.R.-y.-R.; writing—original draft preparation, N.M.-A.; writing—review and editing, E.R.-y.-R. and S.R.-R.; visualization, N.M.-A. and E.R.-y.-R.; supervision, E.R.-y.-R. and S.R.-R.; project administration, E.R.-y.-R. and S.R.-R.; funding acquisition, E.R.-y.-R. and N.M.-A. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by Tecnológico Nacional de México, project number 19211.24-P, and CONAHCYT, scholarship 706358.

**Data Availability Statement:** The data used in this research is available upon request.

**Acknowledgments:** The authors thank Universidad Nacional Autónoma de México and Tecnológico Nacional de México for the concessions granted for the development of the research projects associated with this article.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| BRKGA | Biased Random-Key Genetic Algorithm |
| BKS | Best-known solution |
| CFLNDP | Capacitated Facility Location/Network Problem |
| CFLP | Capacitated Facility Location Problem |
| CLTPI | Capacitated Location Tree Problem |
| CMST | Capacitated minimum spanning tree |
| CRTP | Capacitated Ring-Tree problem |
| EM | Exact method |
| HLP | Hub Location Problem |
| LB | Lower bound |
| LRP | Location-routing problem |
| MILP | Mixed-integer linear programming |
| MPR | Minimum required facilities to open |
| MST | Minimum spanning tree |
| MVA | MegaVolts-Ampers |
| PSO | Particle Swarm Optimization |
| RTFLP | Ring-Tree Facility Location problem |
| THLP | Tree of Hubs Location Problem |
| UB | Upper bound |
| UFLP | Uncapacitated Facility Location Problem |

## References

1. Shaukat, N.; Khan, B.; Ali, S.M.; Mehmood, C.A.; Khan, J.; Farid, U.; Majid, M.; Anwar, S.M.; Jawad, M.; Ullah, Z. A survey on electric vehicle transportation within smart grid system. *Renew. Sustain. Energy Rev.* **2018**, *81*, 1329–1349. [CrossRef]
2. Bayani, R.; Soofi, A.F.; Waseem, M.; Manshadi, S.D. Impact of transportation electrification on the electricity grid—A review. *Vehicles* **2022**, *4*, 1042–1079. [CrossRef]
3. Cicilio, P.; Glennon, D.; Mate, A.; Barnes, A.; Chalishazar, V.; Cotilla-Sanchez, E.; Vaagensmith, B.; Gentle, J.; Rieger, C.; Wies, R.; et al. Resilience in an evolving electrical grid. *Energies* **2021**, *14*, 694. [CrossRef]
4. Brenna, M.; Bucci, V.; Falvo, M.C.; Foiadelli, F.; Ruvio, A.; Sulligoi, G.; Vicenzutti, A. A review on energy efficiency in three transportation sectors: Railways, electrical vehicles and marine. *Energies* **2020**, *13*, 2378. [CrossRef]
5. Kruskal, J.B. On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem. *Proc. Am. Math. Soc.* **1956**, *7*, 48–50. [CrossRef]

6. Prim, R.C. Shortest connection networks and some generalizations. *Bell Syst. Tech. J.* **1957**, *36*, 1389–1401., [CrossRef]
7. Chandy, K.M.; Lo, T. The capacitated minimum spanning tree. *Networks* **1973**, *3*, 173–181. [CrossRef]
8. Papadimitriou, C.H. The complexity of the capacitated tree problem. *Networks* **1978**, *8*, 217–230. [CrossRef]
9. Hakimi, S.L. Optimum distribution of switching centers in a communication network and some related graph theoretic problems. *Oper. Res.* **1965**, *13*, 462–475. [CrossRef]
10. Kuehn, A.A.; Hamburger, M.J. A heuristic program for locating warehouses. *Manag. Sci.* **1963**, *9*, 643–666. [CrossRef]
11. Aráoz, J.; Fernández, E.; Rueda, S. Location routing problems on trees. *Discret. Appl. Math.* **2019**, *259*, 1–18. [CrossRef]
12. Prodhon, C.; Prins, C. A survey of recent research on location-routing problems. *Eur. J. Oper. Res.* **2014**, *238*, 1–17. [CrossRef]
13. Mara, S.T.W.; Kuo, R.; Asih, A.M.S. Location-routing problem: A classification of recent research. *Int. Trans. Oper. Res.* **2021**, *28*, 2941–2983. [CrossRef]
14. Melkote, S.; Daskin, M.S. Capacitated facility location/network design problems. *Eur. J. Oper. Res.* **2001**, *129*, 481–495. [CrossRef]
15. Hill, A.; Voß, S. Optimal capacitated ring trees. *EURO J. Comput. Optim.* **2016**, *4*, 137–166. [CrossRef]
16. Abe, F.H.N.; Hoshino, E.A.; Hill, A. The ring tree facility location problem. *Electron. Notes Discret. Math.* **2015**, *50*, 331–336. [CrossRef]
17. Abe, F.H.; Hoshino, E.A.; Hill, A.; Baldacci, R. A Branch-and-Price Algorithm for the Ring-Tree Facility Location Problem. *Electron. Notes Theor. Comput. Sci.* **2019**, *346*, 3–14. [CrossRef]
18. Contreras, I.; Fernández, E.; Marín, A. The tree of hubs location problem. *Eur. J. Oper. Res.* **2010**, *202*, 390–400. [CrossRef]
19. Ahuja, R.K.; Orlin, J.B.; Sharma, D. Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. *Math. Program.* **2001**, *91*, 71–97. [CrossRef]
20. Ruiz, E.; Albareda-Sambola, M.; Fernández, E.; Resende, M.G. A biased random-key genetic algorithm for the capacitated minimum spanning tree problem. *Comput. Oper. Res.* **2015**, *57*, 95–108. [CrossRef]
21. Ruiz, H.E.R. The Capacitated Minimum Spanning Tree Problem. Ph.D. Thesis, Universitat Politècnica de Catalunya (UPC), Barcelona, Spain, 2013.
22. Holmberg, K.; Rönnqvist, M.; Yuan, D. An exact algorithm for the capacitated facility location problems with single sourcing. *Eur. J. Oper. Res.* **1999**, *113*, 544–559. [CrossRef]
23. Chudak, F.A.; Williamson, D.P. Improved approximation algorithms for capacitated facility location problems. In Proceedings of the International Conference on Integer Programming and Combinatorial Optimization, Graz, Austria, 9–11 June 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 99–113.
24. Rahmani, A.; MirHassani, S. A hybrid firefly-genetic algorithm for the capacitated facility location problem. *Inf. Sci.* **2014**, *283*, 70–78. [CrossRef]
25. Contreras, I.; Fernández, E.; Marín, A. Tight bounds from a path based formulation for the tree of hub location problem. *Comput. Oper. Res.* **2009**, *36*, 3117–3127. [CrossRef]
26. de Sá, E.M.; de Camargo, R.S.; de Miranda, G. An improved Benders decomposition algorithm for the tree of hubs location problem. *Eur. J. Oper. Res.* **2013**, *226*, 185–202. [CrossRef]
27. Wolsey, L.A. *Integer Programming*; John Wiley & Sons: Hoboken, NJ, USA, 2020.
28. Holland, J.H. Genetic algorithms. *Sci. Am.* **1992**, *267*, 66–73. [CrossRef]
29. Van Laarhoven, P.J.; Aarts, E.H.; van Laarhoven, P.J.; Aarts, E.H. *Simulated Annealing*; Springer: Berlin/Heidelberg, Germany, 1987.
30. Dorigo, M.; Gambardella, L.M. Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Trans. Evol. Comput.* **1997**, *1*, 53–66. [CrossRef]
31. Rashedi, E.; Nezamabadi-Pour, H.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
32. Glover, F. Tabu search: A tutorial. *Interfaces* **1990**, *20*, 74–94. [CrossRef]
33. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
34. Yeh, C.C.; Chien, Y.C. A particle swarm optimization-like algorithm for constrained minimal spanning tree problems. *J. Mar. Sci. Technol.* **2014**, *22*, 8.
35. Bean, J.C. Genetic Algorithms and Random Keys for Sequencing and Optimization. *ORSA J. Comput.* **1994**, *6*, 154–160. [CrossRef]
36. Gonçalves, J.F.; Resende, M.G.C. Biased random-key genetic algorithms for combinatorial optimization. *J. Heuristics* **2011**, *17*, 487–525. [CrossRef]
37. Zudio, A.; da Silva Costa, D.H.; Masquio, B.P.; Coelho, I.M.; Pinto, P.E.D. BRKGA/VND hybrid algorithm for the classic three-dimensional bin packing problem. *Electron. Notes Discret. Math.* **2018**, *66*, 175–182. [CrossRef]
38. Ruiz, E.; Soto-Mendoza, V.; Barbosa, A.E.R.; Reyes, R. Solving the open vehicle routing problem with capacity and distance constraints with a biased random key genetic algorithm. *Comput. Ind. Eng.* **2019**, *133*, 207–219. [CrossRef]
39. Lima, A.; Lima, A.; Nogueira, B.; Santos, M.; Pinheiro, R.G. A multi-population brkga for the automatic clustering problem. In Proceedings of the 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Melbourne, Australia, 17–20 October 2021; pp. 368–373.

40. Carballo, P.; Perera, P.; Rama, S.; Pedemonte, M. A Biased Random-Key Genetic Algorithm for Regression Test Case Prioritization. In Proceedings of the 2018 IEEE Latin American Conference on Computational Intelligence (LA-CCI), Guadalajara, Mexico, 7–9 November 2018; pp. 1–6. [CrossRef]
41. Abreu, L.R.; Tavares-Neto, R.F.; Nagano, M.S. A new efficient biased random key genetic algorithm for open shop scheduling with routing by capacitated single vehicle and makespan minimization. *Eng. Appl. Artif. Intell.* **2021**, *104*, 104373. [CrossRef]
42. Toso, R.F.; Resende, M.G. A C++ application programming interface for biased random-key genetic algorithms. *Optim. Methods Softw.* **2015**, *30*, 81–93. [CrossRef]
43. Silva, R.M.A.; Resende, M.G.C.; Pardalos, P.M. A Python/C++ library for bound-constrained global optimization using a biased random-key genetic algorithm. *J. Comb. Optim.* **2015**, *30*, 710–728. [CrossRef]
44. Andrade, C.E.; Toso, R.F.; Gonçalves, J.F.; Resende, M.G. The multi-parent biased random-key genetic algorithm with implicit path-relinking and its real-world applications. *Eur. J. Oper. Res.* **2021**, *289*, 17–30. [CrossRef]
45. Oliveira, B.A.; Xavier, E.C.; Borin, E. BrkgaCuda 2.0: A framework for fast biased random-key genetic algorithms on GPUs. *Soft Comput.* **2024**, *28*, 12689–12704. [CrossRef]
46. Biajoli, F.L.; Chaves, A.A.; Lorena, L.A.N. A biased random-key genetic algorithm for the two-stage capacitated facility location problem. *Expert Syst. Appl.* **2019**, *115*, 418–426. [CrossRef]
47. Souto, G.; Morais, I.; Faulhaber, L.; Ribeiro, G.M.; Henrique González, P. A Hybrid BRKGA Approach for the Two Stage Capacitated Facility Location Problem. In Proceedings of the 2021 IEEE Congress on Evolutionary Computation (CEC), Kraków, Poland, 28 June–1 July 2021; pp. 2007–2014. [CrossRef]
48. Morais, I.; Souto, G.; Ribeiro, G.M.; Mendonça, I.; González, P.H. A hybrid BRKGA approach for the multiproduct two stage capacitated facility location problem. In Proceedings of the 2022 IEEE Congress on Evolutionary Computation (CEC), Padua, Italy, 18–23 July 2022; pp. 1–8.
49. Pessoa, L.S.; Santos, A.C.; Resende, M.G. A biased random-key genetic algorithm for the tree of hubs location problem. *Optim. Lett.* **2017**, *11*, 1371–1384. [CrossRef]
50. Chen, A.l.; Yang, G.k.; Wu, Z.m. Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. *J. Zhejiang-Univ.-Sci. A* **2006**, *7*, 607–614. [CrossRef]
51. MirHassani, S.; Abolghasemi, N. A particle swarm optimization algorithm for open vehicle routing problem. *Expert Syst. Appl.* **2011**, *38*, 11547–11551. [CrossRef]
52. Marinakis, Y.; Marinaki, M.; Dounias, G. A hybrid particle swarm optimization algorithm for the vehicle routing problem. *Eng. Appl. Artif. Intell.* **2010**, *23*, 463–472. [CrossRef]
53. Sevkli, M.; Guner, A.R. A continuous particle swarm optimization algorithm for uncapacitated facility location problem. In Proceedings of the International Workshop on ant Colony Optimization and Swarm Intelligence, Brussels, Belgium, 4–7 September 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 316–323.
54. Gopakumar, A.; Jacob, L. Localization in wireless sensor networks using particle swarm optimization. In Proceedings of the 2008 IET International Conference on Wireless, Mobile and Multimedia Networks, Beijing, China, 11–12 January 2008; pp. 227–230.
55. Onwunalu, J.E.; Durlofsky, L.J. Application of a particle swarm optimization algorithm for determining optimum well location and type. *Comput. Geosci.* **2010**, *14*, 183–198. [CrossRef]
56. Marinakis, Y.; Marinaki, M. A particle swarm optimization algorithm with path relinking for the location routing problem. *J. Math. Model. Algorithms* **2008**, *7*, 59–78. [CrossRef]
57. Marinakis, Y. An improved particle swarm optimization algorithm for the capacitated location routing problem and for the location routing problem with stochastic demands. *Appl. Soft Comput.* **2015**, *37*, 680–701. [CrossRef]
58. Pan, Q.K.; Tasgetiren, M.F.; Liang, Y.C. A discrete particle swarm optimization algorithm for the no-wait flowshop scheduling problem. *Comput. Oper. Res.* **2008**, *35*, 2807–2839. [CrossRef]
59. Kashan, A.H.; Karimi, B. A discrete particle swarm optimization algorithm for scheduling parallel machines. *Comput. Ind. Eng.* **2009**, *56*, 216–223. [CrossRef]
60. Unler, A.; Murat, A. A discrete particle swarm optimization method for feature selection in binary classification problems. *Eur. J. Oper. Res.* **2010**, *206*, 528–539. [CrossRef]
61. Strasser, S.; Goodman, R.; Sheppard, J.; Butcher, S. A new discrete particle swarm optimization algorithm. In Proceedings of the Genetic and Evolutionary Computation Conference 2016, Denver, CO, USA, 20–24 July 2016; pp. 53–60.
62. Jain, M.; Saihjpal, V.; Singh, N.; Singh, S.B. An overview of variants and advancements of PSO algorithm. *Appl. Sci.* **2022**, *12*, 8392. [CrossRef]
63. Shami, T.M.; El-Saleh, A.A.; Alswaitti, M.; Al-Tashi, Q.; Summakieh, M.A.; Mirjalili, S. Particle swarm optimization: A comprehensive survey. *IEEE Access* **2022**, *10*, 10031–10061. [CrossRef]
64. Arumugam, M.S.; Rao, M. On the performance of the particle swarm optimization algorithm with various inertia weight variants for computing optimal control of a class of hybrid systems. *Discret. Dyn. Nat. Soc.* **2006**, *2006*, 079295. [CrossRef]

65. Ozcan, E.; Mohan, C.K. Analysis of a simple particle swarm optimization system. *Intell. Eng. Syst. Through Artif. Neural Netw.* **1998**, *8*, 253–258.
66. Noa Vargas, Y.; Chen, S. Particle swarm optimization with resets—Improving the balance between exploration and exploitation. In Proceedings of the Advances in Soft Computing: 9th Mexican International Conference on Artificial Intelligence, MICAI 2010, Pachuca, Mexico, 8–13 November 2010; Proceedings, Part II 9; Springer: Berlin/Heidelberg, Germany, 2010; pp. 371–381.