sustainability

MDPI

Article

An Improved Cuckoo Search for a Patient Transportation Problem with Consideration of Reducing Transport Emissions

Liyang Xiao [1], Mahjoub Dridi [1], Amir Hajjam El Hassani [1], Hongying Fei [2,*] and Wanlong Lin [3,*]

[1] Nanomedicine Lab, Univ. Bourgogne Franche-Comté, UTBM, 25030 Besancon, France; liyang.xiao@utbm.fr (L.X.); mahjoub.dridi@utbm.fr (M.D.); amir.hajjam-el-hassani@utbm.fr (A.H.E.H.)
[2] School of Management, Shanghai University, 99 ShangDa Road, BaoShan District, Shanghai 200444, China
[3] Shanghai No.3 Rehabilitation Hospital, 100 JiaoCheng Road, JingAn District, Shanghai 200072, China
* Correspondence: Feihy@shu.edu.cn (H.F.); 13162638165@163.com (W.L.)

Received: 26 January 2018; Accepted: 7 March 2018; Published: 13 March 2018

Abstract: Many government agencies and business organizations have realized that it is necessary to consider not only the economic cost but also the road transport emissions when they determine the transport policies and operations. In this study, a patient transportation problem with the aim of reducing transport emissions has been formulated by implementing CVRP model. In order to determine the routes of patient transportation with optimized emissions for targeted hospital, an improved Cuckoo Search (ICS) algorithm is proposed. In this study, a 'split' procedure has been implemented to simplify the individual's representation. A new category of cuckoos has been introduced to improve the ICS's search ability. Two heuristics have been applied to improve the quality of initial population. A local search mechanism has been embedded in the search procedure to improve the quality of solutions obtained at the end of each iteration. The computational results were encouraging and demonstrated the effectiveness of the proposed solution method.

Keywords: vehicle routing; transport emissions; patient transportation

1. Introduction

The transportation sector is one of the main sources of urban noise [1] and greenhouse gases (GHG), which respectively have significant effects on noise and air pollution. According to United States Environmental Protection Agency, transportation sector contributed 27.5% of national GHG emissions in 2015 [2]. With the rapid increase of vehicle numbers in China, road transport emissions have a negative impact air quality and lead to critical environmental and health issues, especially in urban areas. Except CO_2, road transport also generates pollution materials as CO, N_2O, NH_3, CH_4. Reducing road transport emissions has attracted attentions. Many government agencies and business organizations have realized that it is necessary to consider not only the economic cost but also the road transport emissions when they determine the transport policies and operations.

The Vehicle Routing Problem (VRP), in which a set of routes must be defined for a fleet of vehicles to travel from their depot(s) to customers so as to minimize total travel cost or fulfill some other objectives while taking into account a set of given constraints. Many variants of VRPs have been developed to tackle various constraints, such as Capacitated VRP (CVRP), VRP with Time Windows (VRPTW), Multiple Depot VRP (MDVRP) an so on [3]. The VRP not only plays an important role in industrial production but also gets widely applied to other areas [4], such as Supply Chain Logistics [5], Emergency Preparedness [6], Green Logistics [7] and Patient Transportation [8].

Motivated by a real-life problem, this study addresses a patient transportation problem provided by the Shanghai No. 3 Rehabilitation Hospital with consideration of reducing transport emissions. The patient transportation service provided by the targeted hospital transfers its inpatients to other hospitals (medical units) using a fleet of homogeneous vehicles with limited capacity of each. Differing from other health care services, rehabilitation services are all non-emergency, the patient transportation demands are reserved in advance, which are launched by doctors usually according to the medical service requirements of inpatients and the available medical resources among the medical units. This study aims at optimizing the total transport emissions of vehicle routes meanwhile satisfying the patient transportation demands. The research problem can be formulated by implementing CVRP model with consideration of reducing transport emissions. Moreover, in order to solve the problem efficiently with solution of good quality, we then proposed an improved Cuckoo Search (ICS), based on the one developed for solving the famous Travelling Salesman Problem (TSP). The contributions of this study can be concluded as follows: (1) Extended the Cuckoo Search (CS) algorithm, originally designed for TSP, to solve a patient transportation problem with consideration of reducing transport emissions; (2) Implemented a 'split' procedure to simply the individual's representation and local search mechanism; (3) Improved the quality of initial population of CS by using Saving Method and RNNH; (4) Accelerated the efficiency of the proposed ICS by both introducing a new category of cuckoos and using a new local search strategy.

The rest of this paper is organized as follows. First, the relevant literature is reviewed in Section 2; second, details about the patient transportation problem with consideration of reducing transport emissions will be described in Section 3; then, a brief introduction to the basic structure of CS algorithm together with the CS variant proposed by [9] will be given in Section 4. Section 5 will be dedicated to the description of the structure of the proposed CS with its key elements. Afterwards, computational results will be detailed in Section 6 for demonstrating effectiveness and efficiency. This paper will finish with conclusions and perspectives.

2. Literature Review

As a variant of VRP, CVRP is also NP-hard and its computational complexity increases exponentially as the number of customers grows. Although exact methods can obtain the optimal solution, they are not efficient enough, especially for large-size instances [3]. Hence, the requirement to find good solutions quickly (not necessarily the optimal solutions) has led to the development of various heuristic algorithms [10] and approximate algorithms which are also called meta-heuristics in many publications. Some well-structured heuristics can quickly attain feasible solutions for targeted problems. However, the feasible solutions found by heuristic algorithms are not always near the optimal one and thus cannot guarantee the quality of these solutions. On the other hand, a lot of meta-heuristic algorithms, including Particle Swarm Optimization (PSO) [11,12], Tabu Search (TS) [13,14], Simulated Annealing (SA) [15], Genetic Algorithms (GA) [16–18], Squeaky Wheel Optimization (SWO) [19,20] and so on, have been proposed to solve VRPs. According to the literatures, it is easy to apply meta-heuristic algorithms to various VRPs to get solutions with good quality, i.e., solutions quite near to the optimal ones, with acceptable computational time [21–23]. Bio-inspired algorithms are designed after the existing principles in nature systems. In the recent years, it has become an emerging trend in the field of meta-heuristic algorithms to develop bio-inspired algorithms and such algorithms have been widely applied to different scientific and engineering fields [24]. Furthermore, since Thangiah et al. [25] first reported the application of bio-inspired algorithm to the VRP, a large number of proposals have appeared for solving numerous VRP variants [26] with good experimental results [27–30].

It is worth mentioning that one of the bio-inspired meta-heuristic algorithms, developed by Yang and Deb [31] and known as Cuckoo Search (CS), draws researchers' attentions thanks to its strong competence. CS is an optimization algorithm that is inspired by the obligate brood parasitism of some cuckoo species by laying their eggs in the nests of other host birds. According to the literature, the CS algorithm has been applied to continuous optimization problems with better performance than GA

and Particle Swarm Optimization (PSO). Nowadays, CS algorithm and its variants have been applied to many areas with good performance [9,32,33].

Although no variant of the CS algorithm has been developed for solving vehicle routing problem, it was observed that Ouaarab et al. [9] have proposed a discrete CS algorithm for the famous travelling salesman problem (TSP), a special version of VRP problem. Based on a comparison with a set of TSP benchmarks, it was observed that the discrete CS algorithm proposed by Ouaarab et al. [9] outperforms some other popular meta-heuristic algorithms in solving TSP problems. In consequence, we are motivated to develop a CS-based meta-heuristic algorithm to solve the research problem to get satisfactory solutions.

3. Problem Description

Similar to the VRP, the patient transportation problem can also be defined on a graph $G = (V, A)$. The node set $V = \{0, 1, ..., n\}$ where node 0 represents the depot hospital and the other nodes represents other medical units involved in patient transportation service. As for the arc set $A = \{(i, j) \in V\}$, an arc ($(i, j) \in A$) indicates a possible route linking nodes i and j and is associated with a given distance $d_{ij} = d_{ji}$ and velocity $v_{ij} = v_{ji}$. A fleet of K homogeneous vehicles, with limited capacity, are available for transferring the patients from the given depot, indicated as node 0 in set V. Here are some hypotheses:

(1) Each medical unit must be visited by one and only one vehicle;
(2) The trip of each vehicle starts from and ends up at the depot hospital;
(3) The distance and velocity of each arc ($(i, j) \in A$) are constants;
(4) Each medical unit is associated with a given demand d_i of patient transportation service ($i \in V \setminus \{0\}$);
(5) Each vehicle has a limited capacity Q;
(6) Total patients assigned to a vehicle trip must not exceed the vehicle's capacity Q.

Transport Emissions Calculation

The objective is to determine trips of the vehicles visiting all medical units with patient transportation demands to minimize total transport emissions while satisfying all necessary constraints. Speak of transport emissions, MOBILE and COPERT are the most frequently used models [34], which were respectively developed by American and European scientists. There are some other emission models, such as PHEM [35] and TREMOD [36]. To be noticed, both TREMOD and COPERT used Handbook of Emission Factors (HBEFA) database (http://www.hbefa.net/e/index.html). Distance, load and velocity are the most important elements in all these emission models. In this study, COPERT model was used to calculate transport emissions primarily because vehicles in Shanghai currently adopt European-5 standards. The emission factor (EF) [37], which evaluates the quantity of transport emissions exhausted by a single vehicle per kilometer (g/km). The transport emissions of each arc ($(i, j) \in A$), EF_{ij}, can be expressed as:

$$EF_{ij} = (a + c \cdot v_{ij} + e \cdot v_{ij}^2)/(1 + b \cdot v_{ij} + d \cdot v_{ij}^2), \quad (1)$$

where a, b, c, d and e are emission parameters [37].

Let X_{ij}^k be decision variable, which equals to 1 if vehicle trip k travels from node i to node j and otherwise equals to 0. The objection function of the research problem is then formulated in following equation :

$$Min \ \ \Sigma_{i \in V} \Sigma_{j \in V} \Sigma_{k \in K} EF_{ij} d_{ij} X_{ij}^k \quad (2)$$

Subject to

$$\Sigma_{k \in K} \Sigma_{i \in V} X_{ij}^k = 1, \quad j \in \{1, ..., n\} : i \neq j \quad (3)$$

$$\Sigma_{k \in K} \Sigma_{j \in V} X_{ij}^k = 1, \quad i \in \{1, ..., n\} : i \neq j \quad (4)$$

$$\Sigma_{i\in V}\Sigma_{j\in V}X_{ij}^{k}d_i \leq Q, \quad k \in K \tag{5}$$

$$\Sigma_{j\in V}X_{ij}^{k} = \Sigma_{j\in V}X_{ji}^{k}, \quad for\ j \neq 0,\ \ i = 0,\ \ k \in K \tag{6}$$

$$\Sigma_{k\in K}\Sigma_{j\in V}X_{ij}^{k} \leq K, \quad for\ j \neq 0,\ \ i = 0 \tag{7}$$

Objective Function (2) minimizes the total transport emissions. Constraints (3) and (4) make sure that each medical unit is visited by exactly one vehicle. Constraints (5) guarantee that the patients assigned to a trip does not exceed the vehicle's capacity Q. Constraints (6) indicate that the depot hospital is the start and end node for vehicle trips. Constraints (7) guarantee that there are at most K vehicles available for patient transportation.

In order to solve the research problem, we proposed an improved Cuckoo Search. To the best of our knowledge, no existing research has dealt with patient transportation with consideration of reducing transport emissions, so there are no corresponding benchmarks to test the effectiveness of the proposed ICS. By observing Equations (1) and (2), it can be easily concluded that the transport emissions are in direct proportion to total travel distance in the research problem. To evaluate the performance of the ICS proposed in this work, the patient transportation model is reduced into a classical CVRP. As mentioned above, distance is one of the most important element of transport emissions. However, the total travel distance may vary significantly among all the enormous number of feasible routes. Take instance "P-n19-k2" [38] as example, two similar feasible routes are listed in Table 1. Comparing to the second route, the first one reduces 42.57% travel distance and therefore reduces 42.57 % emissions (using COPERT model).

Table 1. Case study: emission comparison between two feasible routes.

Route	Distance
Trip 1: Depot-> 4-> 14-> 11-> 12-> 3->17-> 16-> 6-> 8-> Depot Trip 2: Depot-> 5-> 18-> 13-> 15-> 9-> 7-> 2-> 1-> 10-> Depot	288
Trip 1: Depot-> 4-> 11-> 14-> 12-> 3->17-> 16-> 8-> 6-> Depot Trip 2: Depot-> 18-> 5-> 13-> 15-> 9-> 7-> 2-> 10-> 1-> Depot	202

For a better understanding, we will first introduce the basic idea of CS algorithm. Afterwards, the algorithm developed for TSP [9] (called CS-Quarrab hereinafter) is presented as well as the proposed improved Cuckoo Search algorithm (called ICS in the rest of this paper).

4. Development of the Improved Cuckoo Search

In order to make CS algorithm cope with the TSP [9], where the coordinates of cities are fixed and solutions are presented in the visiting order, the flights of cuckoos in search space should be translated into permutations of visiting order of existing solutions. As mentioned in Section 1, it is observed that the encouraging results in solving TSP with an extension of CS algorithm were published [9], in this paper, we first extended the CS [9] to the vehicle routing problem and then developed an improved CS algorithm to improve not only the quality but also the efficiency of the solution.

4.1. Standard Cuckoo Search

Firstly proposed by Yang and Deb in 2009 [31], the standard Cuckoo Search algorithm (CS) is a meta-heuristic algorithm inspired by the interesting parasitism of cuckoo species and originally developed for solving multimodal functions. CS algorithm can be summarized as three ideal rules: (1) The egg, laid by each cuckoo in the randomly selected nest, represents a random solution; (2) At the end of each iteration (generation), the best nest with an egg of high quality is kept for the next generation, in other words, the solution with the best fitness is preserved; (3) The number of available host nests is fixed, and the cuckoo's egg might be found by the host bird with certain probability

$P_a \in [0, 1]$. If the egg is found, the host bird will throw out the alien egg or abandon its nest so that the egg will not be hatched. In the CS Algorithm, this phenomenon can be described in an easier way that a fraction P_a of the current set of solutions is replaced by randomly generated solutions. A solution X_i^{t+1} is generated from solution X_i^t of cuckoo i by performing a Lévy flight:

$$X_i^{t+1} = X_i^t + \alpha \oplus Levy(s, \lambda) \tag{8}$$

where $\alpha < 0$ is the step size, which should be associated with the scales of the problem of interests and $\alpha = 1$ is most the common used value in the majority of cases.

The most important characteristic of Lévy flights is its intensive search around a solution and the occasional big steps of Lévy flights can minimize the probability of falling into local optima. In fact, the Lévy flight is modelled as a probability density function that has a power law tail, and the step length is associated to the value generated by Lévy flights. Both step length and step size s are randomly drawn from Lévy distribution:

$$Levy(s, \lambda) \sim s^{-\lambda}, \quad (1 < \lambda \leq 3) \tag{9}$$

4.2. CS-Ouaarab Algorithm

Ouaarab et al. [9] proposed a variant of CS algorithm by considering a group of cuckoos performing Lévy flights and adapted this algorithm to solving TSP problem with encouraging results. This extension of CS, denoted as CS-Ouaarab in the rest of this paper, can be described as in Figure 1.

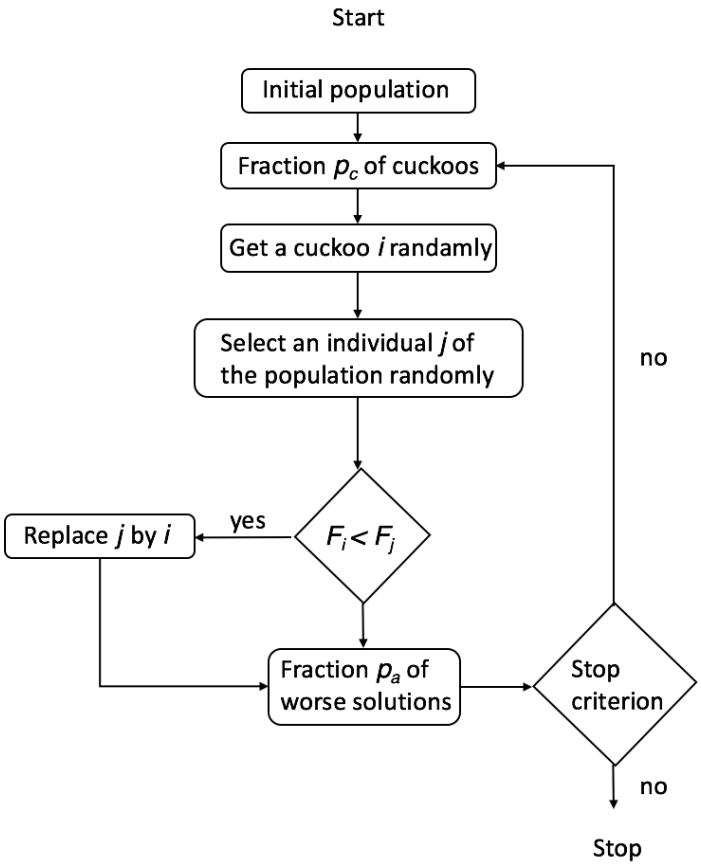

Figure 1. Flowchart of CS-Ouaarab.

5. Improved Cuckoo Search Algorithm

In order to improve the efficiency of CS algorithm and to apply the CS algorithm to tackle CVRP, another variant of CS algorithm is proposed in this study.

Notation:

- Nest: an individual in the population. In this study, a nest contains only one egg;
- Egg: an egg in a nest represents a solution. In this study, a solution is coded as a giant tour, i.e., A sequence of customers without trips delimiters.

Based on the procedure of CS-Quaarab, two improvements are listed as follows:

(1) A local search strategy is performed by a small category of cuckoos with fraction P_d around current nests.

Studies show that a host bird will simply throw the alien eggs or even abandon its nest if it realizes the eggs are not its own. To reduce the probability of their eggs being discovered, some cuckoo species have evolved in such a way that they can engage a kind of surveillance on nests likely to be a host [39]. Furthermore, some female parasitic cuckoos are often very specialized in the mimicry in colour and pattern of the eggs of the chosen host birds so as to reduce the risk of being distinguished. Some other cuckoos observe the hosts around to find whether the chosen nest is the best or not. In consequence, we are inspired to introduce a local search strategy performed by a small category of astute cuckoos with fraction P_d around the current nest.

In the adaptation of CS to TSP, 2-opt [40] and double-bridge moves [41] are performed via Lévy flights to find a new area, where 2-opt is used for the small permutation and large permutations are made by double-bridge moves. In order to apply CS to CVRP, this mechanism is remained in this study: an "astute" cuckoo randomly chooses a direction from its current nest to search for the best nest in a restrictive range of its current nest. To find a better nest, the rule *Reinsertion* [42] will be applied that a randomly chosen node is replaced by the best solution found by the local search, i.e., when a better nest is found, the "astute cuckoo" will abandon its current nest and move forward to the best nest found via local search to improve the hatch condition of its egg. In this study, the moves carried by cuckoos in the search space via Lévy flights are designed to be drawn from the interval [0, 1] as detailed in Table 2.

(2) Set the probability P_b that a cuckoo be "astute" so as to balance between intensification and diversification of solution obtained by the improved CS algorithm.

It is obvious that the new cuckoo category is introduced to strengthen intensive search around their current solutions via best-improvement local search, but our preliminary experiments show that this best-improvement local search strategy of astute cuckoos may cause stagnation in local optima. A probability of astute cuckoos, P_b, is thus proposed to give better resistance against any potential traps and stagnation in local optima. P_b represents the appearance probability of astute cuckoos in each generation which can be regarded as mutation rate in genetic algorithm. The new search mechanism, with a fraction P_d and probability P_b, plays an important role in controlling the balance between intensification and diversification and can be directly introduced in CS.

Table 2. Moves in search space via Lévy flights.

Value of Lévy Flights (Step Length)	Moves Carried by Cuckoos
[**0**, **0.2**)	one 2-opt move
[**0.2**, **0.4**)	two successive 2-opt moves
[**0.4**, **0.6**)	three successive 2-opt moves
[**0.6**, **0.8**)	four successive 2-opt moves
[**0.8**, **1**]	a double-bridge move

The population of the improved CS algorithm (ICS) is associated with three types of Cuckoos:

(1) A fraction P_a of cuckoos that seek for new nests at the end of each generation;
(2) A fraction P_c of cuckoos that search for new nests from the current position. The range of their movements is in proportional to the step length of Lévy flight;
(3) A fraction P_d of astute cuckoos that search with a probability P_b for a better nest in a restrictive range of their current one towards a random direction. Once some better nests are found, the cuckoo will abandon the current nest and move forward to the best nest found. The general procedure of ICS is described as bellow (Algorithm 1):

Algorithm 1: General procedure of ICS

```
Generate initial population of n nests ;
while (t ≤ MaxGeneration) or (not stop criterion) do
    Start searching with a fraction P_c of cuckoos;
    Get a cuckoo randomly by Lévy flights for a new solution (say, i);
    Evaluate its quality/fitness F_i;
    Choose a nest among n (say, j) randomly;
    if F_i > F_j then
        Replace j by the new solution i ;
    end
    Get a random value P in [0, 1];
    if P < P_b then
        Start searching with a fraction P_d of astute cuckoo ;
        Get a cuckoo randomly;
        Replace its current nest (say, k) by the new nest it found (say, l)
    end
    A fraction P_a of worse nests are abandoned and new nests are built;
    Rank the solutions and find the current best;
end
```

5.1. Fitness Evaluation

In ICS, the CS algorithm proposed in this study, a solution is firstly encoded as a giant tour, i.e., a tour without tip delimiters. Then, the giant tour is optimally split into a set of trips with *Split* procedure proposed in [17]. In the *Split* procedure, copies of the depot node (Node 0) are inserted into the giant tour as trip delimiters. This kind of thought was inspired by the route-first, cluster-second heuristic algorithm put forward [43]. Below is an example for better understanding: the optimal route of the instance "P-n19-k2" [38] contains two trips (*Trip1*: *0 4 11 14 12 3 17 16 8 6 0*; *Trip2*: *0 18 5 13 15 9 7 2 10 1 0*). In ICS, the optimal route of "P-n19-k2" is firstly encoded as a giant tour: *4 11 14 12 3 17 16 8 6 18 5 13 15 9 7 2 10 1*. Then, the *Split* procedure loops each subsequence (T_i, T_{i+1}, . . . , T_j) of the given giant tour $T = (T_1, T_2, \ldots, T_m)$ to evaluate whether the trip ($0\ \ T_i, T_{i+1}, \ldots, T_j\ \ 0$) is feasible (not exceeding vehicle capacity Q) or not. Every feasible trip is denoted as one arc $(i-1, j)$ with route cost in order to get the optimal splitting from T_1 to T_m. Therefore, the fitness of a solution is defined as the total travel cost of all trips constructing the corresponding route.

5.2. Host Nest Initialization

To generate a set of initial solutions (nests), one of the most common heuristic algorithms is the saving method [44]. Prins [45] used Greedy Randomized Adaptive Search Procedure (GRASP) [46] and its hybrids using a randomized version of the nearest neighbour heuristic which called RNNH. The initial individuals generated by heuristic methods are expected to evolve to high-quality solutions in a relatively small number of generations of meta-heuristic algorithms [45].

In this study, the initial solutions (host nests) are generated as follows:

- First, generate ten initial solutions with well-known heuristics: two from the savings method (sequential version and parallel version) and eight from RNNH.
- Second, the initial individuals received from the heuristic methods are applied to a 3-opt local search, because Laporte et al. [10] concluded that fairly good solutions can be obtained by adding a 3-opt local search as post-optimization.
- Afterwards, solutions obtained by 3-opt local search are accepted to be part of the initial population.
- Finally, randomly generate the rest individuals of the initial population as giant tours.

6. Numerical Study

To assess the performance of ICS algorithm, three numerical experiments were designed with benchmark problems extracted from site http://neo.lcc.uma.es/vrp/vrp-instances/capacitated-vrp-instances/. ICS algorithm and CS-Quaarab algorithm were both coded with MATLAB 8.6 and executed on a laptop with Core i7 CPU 2.80 GHz.

6.1. Analyses on the Effect of Parameter P_d and P_b

Besides the parameters that adopted from the proposition in [9], it is necessary to tune the value of parameters P_d and P_b which are newly introduced in ICS when developing a new category of cuckoos to improve the quality of initial population. As stated above, the parameters P_d and P_b may have a significant effect over the performance of ICS because the parameter P_b will help to avoid getting trapped in local optima while parameter P_d represents the portion of astute cuckoos in the population.

The tests of parameters are organized in two steps:

Step 1: Evaluation of parameter P_d

Suppose that astute cuckoos appear in every iteration, i.e., ICS with six different values of P_d: 0, 0.05, 0.1, 0.15, 0.2, 0.25, are tested on a eight benchmark problems (2 instances from each A, B, P, E sets) for 500 iterations.

According the experimental results (as shown in Table 3), it was observed that the bigger the value of P_d is, the more execution time is required by the algorithm. With a further analyses of deviation of solutions obtained by ICS with different values of P_d to BKS, i.e., The best known solution (as shown in Figures 2 and 3), it can be concluded that P_d = 0.05 generally has a better performance than the other values over the test instances.

Step 2: Evaluation of Parameter P_b

According to the results in Step 1, P_d is set as 0.05 to obtain a better output of ICS. Similar experiments are designed to evaluate the performance of ICS with four different values of P_b: 0.25, 0.5, 0.75 and 1. According to the experimental results (as shown in Table 3 and 4, Figures 4 and 5), it is obvious that it is better to set P_b = 0.25 to get a better balance between the solution quality (both %Dev(Best) and %Dev(Ave)) and the computational time (show in Figure 6).

Table 3. Results for preliminary experiments done for determining the value of P_d (500 Generations).

Problem	BKS	Results without P_d		Results with P_d									
				P_d = 0.05		P_d = 0.1		P_d = 0.15		P_d = 0.2		P_d = 0.25	
		Best	Time (s)	Best	Time (s)	Best	Time (s)	Best	Time (s)	Best	Time (s)	Best	Time (s)
P-n16-k8	450	450	1.72	450	2.10	450	2.65	450	3.24	450	3.58	450	4.04
P-n23-k8	529	535	2.21	534	3.16	534	4.12	535	5.09	534	5.79	534	6.64
B-n35-k5	955	967	3.28	960	5.07	963	6.70	962	8.42	967	9.89	962	11.59
B-n43-k6	742	749	4.26	742	6.86	749	9.26	743	11.69	743	14.05	743	16.51
A-n55-k9	1073	1109	6.67	1087	10.98	1090	15.01	1090	18.50	1088	22.96	1090	27.08
A-n69-k9	1174	1191	9.33	1188	15.53	1188	22.14	1188	28.18	1188	33.85	1188	40.60
E-n76-k7	682	717	10.62	715	18.18	717	25.76	717	33.42	717	40.03	717	47.59
E-n76-k8	735	782	10.64	771	18.15	783	25.60	788	33.34	765	39.36	772	46.30

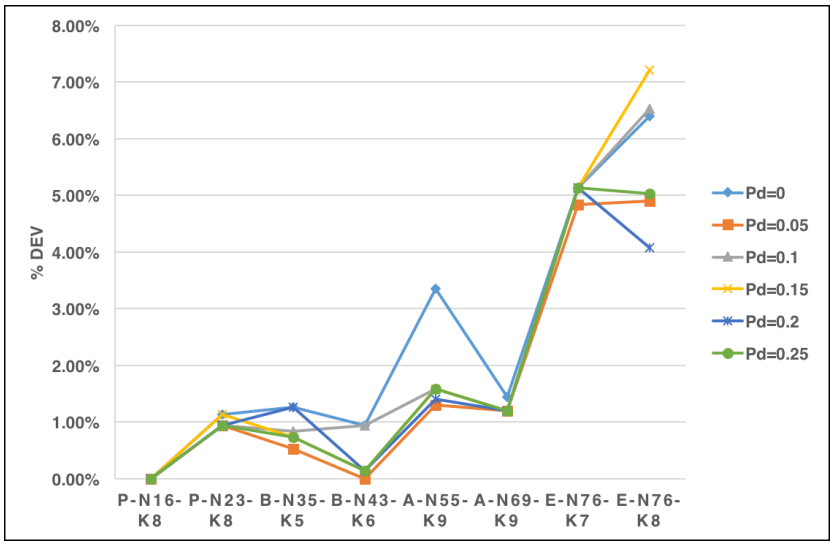

Figure 2. Percentage deviations of best results from the best-known solutions according to different P_d values.

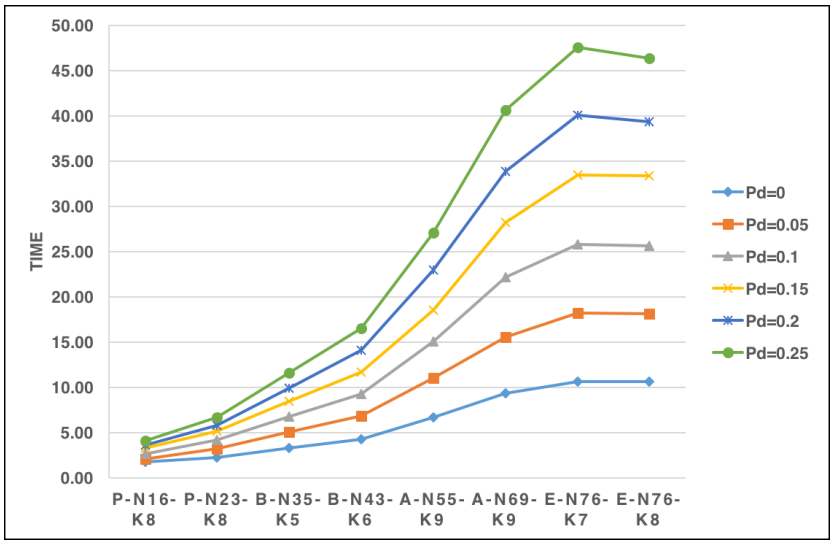

Figure 3. Computational time variations according to different P_d values.

Table 4. Results for preliminary experiments done for determining the value of P_b (500 Generations).

Problem	BKS	P_b = 0.25			P_b = 0.5			P_b = 0.75			P_b = 1		
		Best	Average	Time (s)	Best	Average	Time (s)	Best	Average	Time (s)	Best	Average	Time (s)
P-n16-k8	450	450	450.00	1.72	450	450.00	1.84	450	450.00	1.98	450	451.00	2.11
P-n23-k8	529	534	535.20	2.50	535	536.20	2.69	534	536.40	2.90	534	535.80	3.16
B-n35-k5	955	962	965.00	3.65	962	966.20	4.06	963	964.80	4.51	960	966.40	5.07
B-n43-k6	742	742	746.40	4.94	749	750.20	5.54	742	748.00	6.17	742	749.20	6.86
A-n55-k9	1073	1087	1093.20	7.67	1090	1101.40	8.49	1105	1108.20	9.52	1087	1101.20	10.98
A-n69-k9	1174	1185	1187.40	10.73	1187	1187.80	12.14	1188	1188.60	13.79	1188	1188.00	15.53
E-n76-k7	682	710	718.80	12.95	717	722.00	14.19	718	719.00	15.98	715	721.40	18.18
E-n76-k8	735	757	773.80	12.87	788	788.60	14.29	779	783.00	15.90	771	782.00	18.15

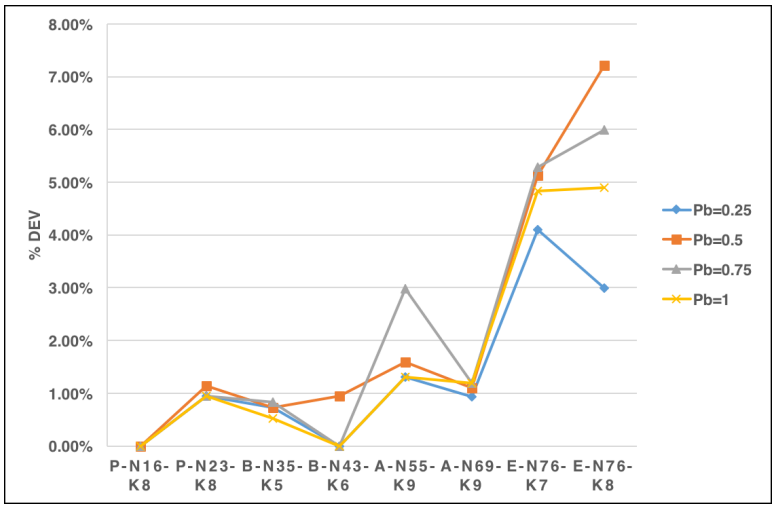

Figure 4. Percentage deviations of best results from the best-known solutions according to different P_b values.

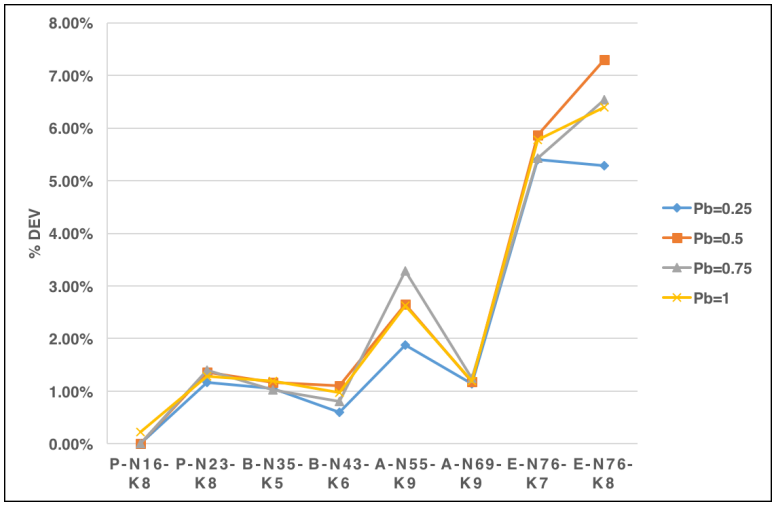

Figure 5. Percentage deviations of average results from the best-known solutions according to different P_b values.

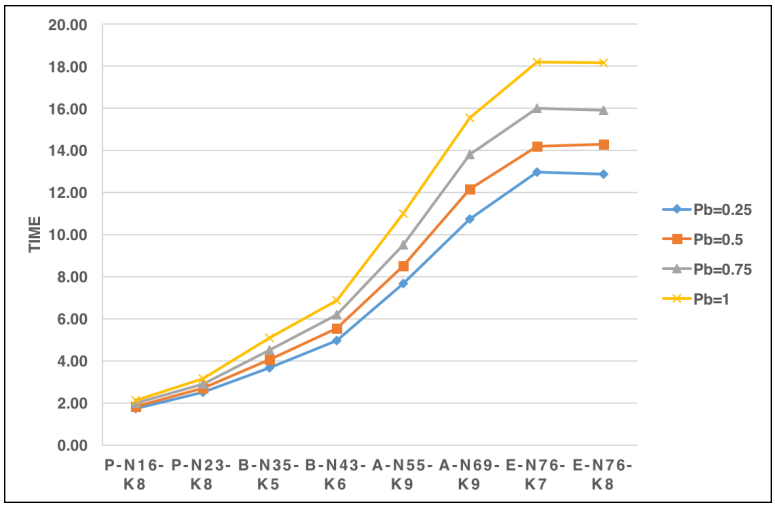

Figure 6. Computational time variations according to different P_b values.

6.2. Comparison between CS-Quaarab and ICS

In this subsection, numerical experiments were performed to make a comparison between the CS-Quarrab algorithm and the proposed ICS on eighty-two well-known CVRP instances, which includes full seventy-four instances from A, B, P sets [38] and eight instances from set E [47]. All the above-mentioned instances have already been optimality solved and the BKS values were derived from [48]. Each instance was executed for 30 runs with parameters given in Table 5.

Table 5. Parameter settings for both algorithms, CS-Ouaarab and ICS.

Parameter	Value	Meaning
n	20	Population size
P_a	0.2	Portion of bad solutions
P_c	0.6	Portion of cuckoos performing Lévy flights
P_d	0.05	Portion of astute cuckoos (only for the improved CS)
P_b	0.25	Probability of astute cuckoos (only for the improved CS)
MaxGeneration	5000	Maximum number of iterations
Stop criterion	1000	Attempt limit of successive iterations

According to the results shown in Tables 6–9, it can be observed that ICS outperforms CS-Quaarab for all instances regarding solution quality. It is worth mentioning that ICS attains BKS for 22 instances (about 27%) while CS-Quarrab just attained BKS for 4 instances (about 4.9%). Although it can be observed that ICS consumes more CPU than CS-Quaarab, however, ICS can obtain the solution within 1 min for most of the instances which means that all instances can be solved in reasonable execution time. The results are quite encouraging.

Table 6. Comparison of both algorithms, the CS-Ouaarab and the ICS on problem set A.

Problem	BKS	CS-Ouaarab						ICS					
		Best	Average	Worst	%Dev( )		Time(s)	Best	Average	Worst	%Dev( )		Time(s)
					Best	Average					Best	Average	
A-n32-k5	784	798	823.03	917	1.79	4.98	6.19	784	787.96	796	0.00	0.51	12.86
A-n33-k5	661	674	710.90	738	1.97	7.55	6.14	661	662.93	677	0.00	0.29	11.98
A-n33-k6	742	756	772.43	805	1.89	4.10	7.82	742	746.43	755	0.00	0.60	14.62
A-n34-k5	778	805	833.60	896	3.47	7.15	6.76	778	784.53	789	0.00	0.84	9.28
A-n36-k5	799	830	865.90	945	3.88	8.37	7.03	799	809.50	815	0.00	1.31	10.52
A-n37-k5	669	695	734.83	780	3.89	9.84	7.39	670	680.26	702	0.15	1.68	14.63
A-n37-k6	949	975	1020.70	1059	2.74	7.56	8.79	958	970.86	977	0.95	2.30	11.99
A-n38-k5	730	744	776.76	823	1.92	6.41	7.26	732	740.00	748	0.27	1.37	13.62
A-n39-k5	822	863	907.70	980	4.99	10.43	8.50	822	833.73	868	0.00	1.43	22.84
A-n39-k6	831	861	887.16	928	3.61	6.76	7.88	834	846.80	857	0.36	1.90	12.56
A-n44-k6	937	978	1039.50	1097	4.38	10.94	9.05	948	955.60	963	1.17	1.99	15.11
A-n45-k6	944	985	1020.20	1114	4.34	8.07	9.90	961	974.06	993	1.80	3.18	20.95
A-n45-k7	1146	1211	1249.10	1299	5.67	9.00	8.42	1161	1194.70	1198	1.31	4.25	12.17
A-n46-k7	914	928	1023.70	1066	1.53	12.00	9.82	915	925.50	939	0.11	1.26	18.82
A-n48-k7	1073	1135	1181.20	1221	5.78	10.08	10.48	1089	1099.60	1110	1.49	2.48	16.89
A-n53-k7	1010	1069	1120.90	1200	5.84	10.98	11.53	1033	1057.50	1098	2.28	4.70	26.61
A-n54-k7	1167	1222	1292.90	1352	4.71	10.79	10.62	1186	1200.70	1209	1.63	2.89	15.51
A-n55-k9	1073	1142	1167.90	1221	6.43	8.84	12.20	1075	1089.80	1109	0.19	1.57	24.94
A-n60-k9	1354	1450	1500.70	1541	7.09	10.83	16.02	1365	1365.00	1365	0.81	0.81	17.21
A-n61-k9	1034	1080	1130.80	1202	4.45	9.36	13.28	1047	1061.10	1072	1.26	2.62	31.62
A-n62-k8	1288	1346	1406.80	1469	4.50	9.22	11.93	1324	1332.70	1341	2.80	3.47	21.21
A-n63-k9	1616	1709	1779.40	1841	5.75	10.11	13.32	1654	1654.80	1655	2.35	2.40	19.42
A-n63-k10	1314	1391	1445.00	1494	5.86	9.97	13.68	1332	1347.50	1348	1.37	2.55	19.24
A-n64-k9	1401	1491	1536.50	1585	6.42	9.67	12.03	1446	1452.40	1455	3.21	3.67	21.68
A-n65-k9	1174	1258	1304.90	1362	7.16	11.15	12.54	1199	1230.00	1265	2.13	4.77	45.05
A-n69-k9	1159	1252	1297.80	1397	8.02	11.98	14.19	1181	1187.10	1188	1.90	2.42	23.99
A-n80-k10	1763	1875	1973.50	2062	6.35	11.94	17.02	1832	1837.70	1840	3.91	4.24	27.33
Average					4.61	9.19	10.36				1.16	2.28	18.99

Table 7. Comparison of both algorithms, the CS-Ouaarab and the ICS on problem set B.

Problem	BKS	CS-Ouaarab						ICS					
		Best	Average	Worst	%Dev( )		Time(s)	Best	Average	Worst	%Dev( )		Time(s)
					Best	Average					Best	Average	
B-n31-k5	672	675	683.36	701	0.45	1.69	5.41	672	672.66	677	0.00	0.10	8.93
B-n34-k5	788	795	826.96	889	0.89	4.94	9.35	789	791.86	793	0.13	0.49	8.18
B-n35-k5	955	973	983.60	1003	1.88	2.99	8.36	958	963.00	967	0.31	0.84	10.74
B-n38-k6	805	822	834.46	848	2.11	3.66	7.88	807	815.03	822	0.25	1.25	16.31
B-n39-k5	549	571	604.33	667	4.01	10.08	9.43	557	562.53	565	1.46	2.46	8.98
B-n41-k6	829	841	866.43	895	1.45	4.52	9.92	831	839.43	858	0.24	1.26	20.29
B-n43-k6	742	748	759.26	796	0.81	2.33	8.02	742	744.86	751	0.00	0.39	17.12
B-n44-k7	909	936	969.46	1003	2.97	6.65	10.57	910	921.26	934	0.11	1.35	15.29
B-n45-k5	751	785	803.63	852	4.53	7.01	9.79	752	752.00	752	0.13	0.13	10.46
B-n45-k6	678	722	739.46	760	6.49	9.06	8.45	690	713.60	715	1.77	5.25	10.76
B-n50-k7	741	760	777.23	819	2.56	4.89	10.42	741	744.60	745	0.00	0.49	13.03
B-n50-k8	1312	1356	1388.70	1431	3.35	5.85	12.08	1331	1344.50	1356	1.45	2.48	21.77
B-n51-k7	1016	1022	1048.80	1103	0.59	3.23	11.12	1016	1018.50	1025	0.00	0.25	26.16
B-n52-k7	747	761	778.40	809	1.87	4.20	10.72	754	755.63	757	0.94	1.16	15.25
B-n56-k7	707	742	760.26	795	4.95	7.53	12.30	720	725.30	728	1.84	2.59	17.58
B-n57-k7	1144	1161	1212.40	1308	1.49	5.98	14.43	1144	1148.30	1164	0.00	0.38	34.46
B-n57-k9	1598	1646	1685.70	1734	3.00	5.49	13.93	1611	1635.20	1650	0.81	2.33	26.35
B-n63-k10	1496	1586	1639.80	1694	6.02	9.61	13.52	1554	1567.40	1582	3.88	4.78	27.47
B-n64-k9	861	917	935.50	963	6.50	8.65	12.32	894	903.30	915	3.83	4.91	30.69
B-n66-k9	1316	1378	1418.70	1464	4.71	7.80	13.90	1331	1350.40	1391	1.14	2.61	42.43
B-n67-k10	1032	1099	1118.00	1179	6.39	8.23	14.27	1073	1084.70	1097	3.87	5.00	42.84
B-n68-k9	1272	1322	1352.30	1416	3.93	6.31	16.79	1288	1302.40	1310	1.26	2.39	31.98
B-n78-k10	1221	1296	1331.50	1387	6.14	9.05	20.68	1257	1261.30	1262	2.95	3.30	26.66
Average					3.35	6.08	11.46				1.15	2.01	21.03

Table 8. Comparison of both algorithms, the CS-Ouaarab and the ICS on problem set E.

Problem	BKS	CS-Ouaarab						ICS					
		Best	Average	Worst	%Dev( )		Time(s)	Best	Average	Worst	%Dev( )		Time(s)
					Best	Average					Best	Average	
E-n22-k4	375	377	390.63	414	0.53	4.17	5.18	375	375.00	375	0.00	0.00	7.15
E-n23-k3	569	570	578.70	623	0.18	1.70	4.20	569	569.50	574	0.00	0.09	6.33
E-n30-k4	503	506	525.56	573	0.60	4.49	6.07	503	504.40	510	0.00	0.28	11.08
E-n33-k4	835	853	887.10	934	2.16	6.24	7.95	837	839.86	841	0.24	0.58	6.95
E-n76-k7	682	742	778.73	812	8.80	14.18	16.84	710	715.36	724	4.11	4.89	40.12
E-n76-k8	735	800	842.73	870	8.84	14.66	13.56	751	765.76	788	2.18	4.19	55.07
E-n76-k14	1021	1100	1130.00	1156	8.84	14.66	13.56	1045	1046.90	1047	2.35	2.54	26.09
E-n101-k14	1067	1197	1230.20	1274	12.18	15.30	22.38	1109	1110.50	1112	3.94	4.08	45.71
Average					5.13	8.93	11.74				1.60	2.08	24.81

Table 9. Comparison of both algorithms, the CS-Ouaarab and the ICS on problem set P.

Problem	BKS	CS-Ouaarab						ICS					
		Best	Average	Worst	%Dev( )		Time(s)	Best	Average	Worst	%Dev( )		Time(s)
					Best	Average					Best	Average	
P-n16-k8	450	450	450.00	450	0.00	0.00	4.53	450	450.00	450	0.00	0.00	4.23
P-n19-k2	212	212	224.36	239	0.00	5.83	3.76	212	212.00	212	0.00	0.00	4.14
P-n20-k2	216	218	228.80	246	0.93	5.93	4.06	216	216.00	216	0.00	0.00	6.06
P-n21-k2	211	219	226.86	257	3.79	7.52	3.74	211	211.41	212	0.00	0.19	5.76
P-n22-k2	216	217	231.33	264	0.46	7.10	3.97	216	216.00	216	0.00	0.00	6.37
P-n22-k8	590	590	594.27	611	0.00	0.72	8.13	590	590.00	590	0.00	0.00	4.56
P-n23-k8	529	529	538.66	550	0.00	1.83	9.74	529	532.41	534	0.00	0.64	7.39
P-n40-k5	458	481	510.06	559	5.02	11.37	7.96	458	458.79	461	0.00	0.17	24.96
P-n45-k5	510	540	565.46	587	5.88	10.87	8.04	512	522.03	528	0.39	2.36	16.66
P-n50-k7	554	578	613.90	644	4.33	10.81	12.08	568	569.96	573	2.53	2.88	25.17
P-n50-k8	631	664	685.66	723	5.23	8.66	12.30	635	642.75	647	0.63	1.86	21.25
P-n50-k10	696	738	756.63	777	6.03	8.71	11.41	712	717.89	721	2.30	3.15	25.50
P-n51-k10	741	786	810.43	840	6.07	9.37	13.72	751	754.37	757	1.35	1.80	26.02
P-n55-k7	568	600	628.96	673	5.63	10.73	12.41	577	582.17	583	1.58	2.49	37.64
P-n55-k8	588	621	640.30	665	7.81	11.16	12.24	584	596.44	599	1.39	3.55	40.15
P-n55-k10	694	729	752.53	775	5.04	8.43	12.98	708	719.27	726	2.02	3.64	23.44
P-n55-k15	945	975	1007.90	1040	3.17	6.66	14.23	955	956.65	957	1.06	1.23	25.09
P-n60-k10	744	807	826.46	841	8.47	11.08	13.18	774	779.82	784	4.03	4.81	35.03
P-n60-k15	968	1013	1051.30	1074	4.65	8.61	16.73	993	1011.40	1014	4.48	4.48	22.46
P-n65-k10	792	862	889.70	923	8.84	12.34	14.08	826	833.44	837	4.29	5.23	24.35
P-n70-k10	827	889	928.60	966	7.50	12.29	14.09	840	851.68	856	1.57	2.98	46.41
P-n76-k4	593	652	684.13	711	9.95	15.37	15.69	618	623.27	627	4.22	5.10	56.43
P-n76-k5	627	667	725.00	765	6.38	15.63	14.29	652	658.31	660	3.99	4.99	68.13
P-n101-k4	681	756	787.30	838	11.01	15.61	21.00	690	711.48	722	1.32	4.48	114.13
Average					4.84	9.03	11.02				1.47	2.34	27.97

6.3. Comparison between ICS and Some Other Recently Published Methods Tackling CVRP

In order to further evaluate the performance of ICS, with the output of ICS is compared with some other recently published methods that tackle CVRPs. Mohammed et al. [49] lately applied K-Nearest Neighbor Algorithm (KNNA) for solving CVRP and show the results of several instances. Although they did not indicate the experimental environment, their results are still listed for the comparison of solution quality. In this paper, three complete benchmarks are selected: Unsupervised Fuzzy Clustering approach (UFC) [50], large Neighbourhood Search algorithm by accepting only the improving solutions (LNSi) [51] and a hybrid algorithm that executes Large Neighbourhood Search algorithm in combination with the solution construction mechanism of the Ant Colony Optimization algorithm (LNS-ACO) [51]. The author indicated that although LNS-ACO outperforms LNSi in solution quality yet LNSi was much faster than LNS-ACO.

These algorithms are chosen for comparison mainly because they are most recently published methods that also reported results for CVRP instances sets A, B, P [38] and E [47]. Moreover, LNSi and LNS-ACO were also coded in Matlab and run the algorithms on a personal computer with Core i7 CPU 2.80 GHz [51]. With parameters given in Table 5, the comparison between the output of ICS and the results of the other algorithms are shown in Tables 10–13 where BKSs are referred to [48].

Table 10. Computational results for the problem set A.

Problem	BKS	KNNA	UFC		LNSi		LNS-ACO		ICS	
		%Dev( )	%Dev( )		%Dev( )		%Dev( )		%Dev( )	
			Worst	Best	Worst	Best	Worst	Best	Worst	Best
A-n32-k5	784	1.66	6.1	3.6	0.00	0.00	0.00	0.00	1.53	0.00
A-n33-k5	661	21.79	4.2	2.9	0.61	0.00	0.00	0.00	2.42	0.00
A-n33-k6	742	17.12	3.0	2.0	0.54	0.00	0.00	0.00	1.75	0.00
A-n34-k5	778	-	2.4	1.5	1.54	0.77	0.00	0.00	1.41	0.00
A-n36-k5	799	-	2.2	2.2	1.13	0.75	0.00	0.00	2.00	0.00
A-n37-k5	669	31.99	7.9	3.6	2.09	1.20	0.00	0.00	4.93	0.15
A-n37-k6	949	6.90	3.3	2.9	1.26	0.42	0.00	0.00	2.95	0.95
A-n38-k5	730	-	8.2	2.4	2.33	1.37	0.00	0.00	2.47	0.27
A-n39-k5	822	-	7.2	3.3	2.55	1.70	0.00	0.00	5.60	0.00
A-n39-k6	831	17.57	4.4	1.7	2.17	1.44	0.24	0.00	3.13	0.36
A-n44-k6	937	21.34	5.9	2.9	2.67	1.49	0.53	0.00	2.77	1.17
A-n45-k6	944	19.60	7.1	1.3	4.34	3.07	2.01	1.48	5.19	1.80
A-n45-k7	1146	-	6.6	4.6	3.05	2.36	0.52	0.00	4.54	1.31
A-n46-k7	914	26.26	5.4	2.4	3.17	1.75	0.33	0.00	2.74	0.11
A-n48-k7	1073	12.12	4.3	2.3	3.17	2.24	1.03	1.03	3.45	1.49
A-n53-k7	1010	20.20	13.7	8.3	3.56	2.38	0.69	0.00	8.71	2.28
A-n54-k7	1167	4.46	6.7	2.7	2.31	1.71	0.60	0.00	3.60	1.63
A-n55-k9	1073	29.26	6.4	3.7	3.08	2.05	0.09	0.00	3.36	0.19
A-n60-k9	1354	6.06	6.3	3.6	2.81	1.92	0.37	0.00	0.81	0.81
A-n61-k9	1034	-	11.9	7.3	6.19	4.93	4.06	3.19	3.68	1.26
A-n62-k8	1288	6.13	8.4	4.5	4.58	3.34	1.94	1.55	4.11	2.80
A-n63-k9	1616	-	7.6	4.1	4.21	2.85	2.35	2.04	2.41	2.35
A-n63-k10	1314	15.00	6.4	2.6	4.72	3.65	1.75	1.14	2.59	1.37
A-n64-k9	1401	-	7.3	4.2	4.07	3.00	1.50	1.00	3.85	3.21
A-n65-k9	1174	25.30	9.6	4.8	4.77	3.24	1.45	0.94	7.75	2.13
A-n69-k9	1159	33.91	8.0	4.4	4.31	3.02	1.64	0.95	2.50	1.90
A-n80-k10	1763	-	8.9	6.5	5.67	4.71	3.35	2.95	4.37	3.91

As show in Table 10, the experimental results of the comparison between ICS and KNNA are given. It can be seen from this table that the solutions obtained by ICS are always much better than those of KNNA. In the best case (A-n37-k5), the gaps between approximate solutions and BKS have been reduced from 31.99% to 0.15%. On average, the gaps have been reduced from 17.59% to 1.06%. According to the summary shown in Table 14, the effectiveness of the proposed ICS has been shown, since it was able to produce best-known results for 22 instances out of 81 and ICS is superior to both UFC and LNSi with higher success rate (SR) for nearly all the testing instance sets A (Table 10),

B (Table 11), E (Table 12), P (Table 13). Based on the detailed information shown in Tables 10–13, it was obvious that the stability of ICS is much better than UFC and LNSi as well. As for LNS-ACO, although it shows better performances in general case in terms of solution quality, ICS is still comparable to LNS-ACO: for about 30.9% (25/81) instances, the solutions obtained by ICS are equal to or better than those obtained by LNS-ACO, where ICS outperforms LNS-ACO in three instances (A-n61-k9, P-n50-k8 and B-n68-k9). For the other instances, the gaps between solutions obtained by these two methods are always less than 3.3%. On the other hand, ICS is the most efficient among all the algorithms. As shown in Table 15, ratios of ICS's computational time to that of LNSi varied from 0.40% to 2.15%. The ratios of ICS to LNS-ACO are even smaller (0.38–2.08%)!

Table 11. Computational results for the problem set B.

Problem	BKS	UFC		LNSi		LNS-ACO		ICS	
		%Dev( )		%Dev( )		%Dev( )		%Dev( )	
		Worst	Best	Worst	Best	Worst	Best	Worst	Best
B-n31-k5	672	1.4	0.6	0.00	0.00	0.00	0.00	0.74	0.00
B-n34-k5	788	3.6	1.8	0.00	0.00	0.00	0.00	0.63	0.13
B-n35-k5	955	3.5	2.7	0.00	0.00	0.00	0.00	1.26	0.31
B-n38-k6	805	6.7	3.8	0.00	0.00	0.00	0.00	2.11	0.25
B-n39-k5	549	8.5	3.5	4.19	2.19	0.00	0.00	2.91	1.46
B-n41-k6	829	3.6	2.7	2.41	1.33	0.24	0.00	3.50	0.24
B-n43-k6	742	6.0	3.4	2.96	1.48	0.40	0.00	1.21	0.00
B-n44-k7	909	2.6	2.2	2.09	1.10	0.00	0.00	2.75	0.11
B-n45-k5	751	5.4	3.0	3.33	1.60	0.00	0.00	0.13	0.13
B-n45-k7	678	8.9	4.6	3.98	2.36	0.29	0.00	5.46	1.77
B-n50-k7	741	7.3	7.2	3.78	2.16	0.00	0.00	0.54	0.00
B-n50-k8	1312	5.1	4.0	2.67	1.75	0.53	0.00	3.35	1.45
B-n51-k8	1016	3.0	2.6	2.36	1.28	0.20	0.00	0.89	0.00
B-n52-k8	747	6.8	2.7	3.35	1.47	0.13	0.00	1.34	0.94
B-n56-k7	707	4.8	2.1	3.54	1.98	0.00	0.00	2.97	1.84
B-n57-k8	1140	9.5	6.3	2.54	1.49	0.00	0.00	1.75	0.00
B-n57-k9	1598	5.0	3.5	2.25	1.13	0.31	0.00	3.25	0.81
B-n63-k10	1496	5.6	4.7	3.68	2.41	1.20	1.20	5.75	3.88
B-n64-k9	861	8.2	5.4	5.92	3.60	2.09	1.51	6.27	3.83
B-n66-k9	1316	3.9	2.2	3.65	2.43	1.52	1.06	5.70	1.14
B-n67-k10	1032	8.9	3.9	5.23	3.59	2.52	1.74	6.20	3.87
B-n68-k9	1272	6.1	3.5	4.09	2.83	1.97	1.42	2.99	1.26
B-n78-k10	1221	7.3	0.0	3.11	1.64	1.23	0.57	3.36	2.95

Table 12. Computational results for the problem set E.

Problem	BKS	UFC		LNSi		LNS-ACO		ICS	
		%Dev( )		%Dev( )		%Dev( )		%Dev( )	
		Worst	Best	Worst	Best	Worst	Best	Worst	Best
E-n22-k4	375	0.1	0.1	0.80	0.00	0.00	0.00	0.00	0.00
E-n23-k3	569	0.3	0.1	0.88	0.00	0.00	0.00	0.88	0.00
E-n30-k4	503	-	-	1.79	0.00	0.00	0.00	1.39	0.00
E-n33-k4	835	2.2	1.2	1.68	0.36	0.00	0.00	0.72	0.24
E-n76-k7	682	8.7	5.8	6.74	4.99	1.91	1.91	6.16	4.11
E-n76-k8	735	7.6	5.4	6.26	4.22	3.13	1.22	7.21	2.18
E-n76-k14	1021	13.6	7.6	5.29	4.21	4.21	0.88	2.55	2.35
E-n101-k14	1067	14.1	10.2	5.25	3.47	1.41	1.41	4.22	3.94

Table 13. Computational results for the problem set P.

Problem	BKS	UFC		LNSi		LNS-ACO		ICS	
		%Dev( )		%Dev( )		%Dev( )		%Dev( )	
		Worst	Best	Worst	Best	Worst	Best	Worst	Best
P-n16-k8	450	1.6	0.2	0.00	0.00	0.00	0.00	0.00	0.00
P-n19-k2	212	3.7	3.7	0.00	0.00	0.00	0.00	0.00	0.00
P-n20-k2	216	1.1	1.1	0.00	0.00	0.00	0.00	0.00	0.00
P-n21-k2	211	14.5	4.3	0.00	0.00	0.00	0.00	0.47	0.00
P-n22-k2	216	0.9	0.9	0.00	0.00	0.00	0.00	0.00	0.00
P-n22-k8	590	12.3	4.6	1.19	0.00	0.00	0.00	0.00	0.00
P-n23-k8	529	-	-	1.70	0.00	0.00	0.00	0.95	0.00
P-n40-k5	458	4.4	2.3	5.02	3.28	0.00	0.00	0.66	0.00
P-n45-k5	510	0.7	0.5	4.51	2.75	0.00	0.00	3.53	0.39
P-n50-k7	554	6.6	4.5	4.87	2.71	0.90	0.00	3.43	2.53
P-n50-k8	631	-	-	5.86	4.28	2.85	1.90	2.54	0.63
P-n50-k10	696	7.9	5.1	3.45	1.87	0.57	0.00	3.59	2.30
P-n51-k10	741	9.5	6.0	4.32	2.70	1.48	0.81	2.16	1.35
P-n55-k7	568	15.0	12.5	4.93	2.11	0.00	0.00	2.64	1.58
P-n55-k8	588	5.7	3.3	3.91	2.04	0.17	0.00	3.99	1.39
P-n55-k10	694	10.9	7.2	2.88	2.02	0.72	0.00	4.61	2.02
P-n55-k15	989	-	-	2.22	1.31	0.00	0.00	1.27	1.06
P-n60-k10	744	14.1	10.2	5.91	3.76	2.15	1.48	5.38	4.03
P-n60-k15	968	16.8	12.3	3.82	2.69	1.55	0.93	4.75	2.58
P-n65-k10	792	8.2	4.1	3.79	2.65	1.77	1.01	5.68	4.29
P-n70-k10	827	11.5	5.9	5.20	2.90	2.06	1.21	3.51	1.57
P-n76-k4	593	5.6	2.8	6.75	3.54	1.69	0.84	5.73	4.22
P-n76-k5	627	4.3	1.6	7.50	4.78	3.67	2.87	5.26	3.99

Table 14. Summary of optimum achievements.

Problem Set	Algorithm											
	UFC			LNSi			LNS-ACO			ICS		
	OA	NI	SR	OA	NI	SR	OA	NI	SR	OA	NI	SR
A	0	27	0%	3	27	11%	17	27	63%	6	27	22%
B	1	23	4%	4	23	17%	16	23	70%	5	23	22%
E	0	7	0%	3	8	38%	4	8	50%	3	8	38%
P	0	20	0%	7	23	30%	15	23	65%	8	23	35%
Overall	1	77	1%	17	81	21%	52	81	64%	22	81	27%

OA: Number of optimum achievements; NI: Number of instances; SR: Success rate (OA/NI).

Table 15. Computational Time Comparison of : LNSi, LNS-ACO and ICS.

Problem	Time(s)			Problem	Time(s)		
	LNSi	LNS-ACO	ICS		LNSi	LNS-ACO	ICS
A-n32-k5	834.81	856.21	12.86	B-n56-k7	2669.15	2709.80	17.58
A-n33-k5	877.56	900.06	11.98	B-n57-k8	2801.73	2844.40	34.46
A-n33-k6	924.16	947.86	14.62	B-n57-k9	3009.96	3055.80	26.35
A-n34-k5	886.31	909.04	9.28	B-n63-k10	3694.64	3750.90	27.47
A-n36-k5	1029.21	1055.60	10.52	B-n64-k9	3777.08	3834.60	30.69
A-n37-k5	1076.11	1103.70	14.63	B-n66-k9	4178.96	4242.60	42.43
A-n37-k6	1085.37	1113.40	11.99	B-n67-k10	4455.16	4523.00	42.84
A-n38-k5	1110.72	1139.20	13.62	B-n68-k9	4329.27	4395.20	31.98
A-n39-k5	1172.73	1202.80	22.84	B-n78-k10	5958.46	6049.20	26.66
A-n39-k6	1234.35	1266.00	12.56	E-n22-k4	425.02	447.39	7.15
A-n44-k6	1528.61	1567.80	15.11	E-n23-k3	373.42	393.08	6.33

Table 15. *Cont.*

Problem	Time(s)			Problem	Time(s)		
	LNSi	LNS-ACO	ICS		LNSi	LNS-ACO	ICS
A-n45-k6	1648.90	1728.10	20.95	E-n30-k4	664.03	698.98	11.08
A-n45-k7	1698.16	1741.70	12.17	E-n33-k4	777.13	818.03	6.95
A-n46-k7	1759.39	1804.50	18.82	E-n76-k7	4926.35	5186.30	40.12
A-n48-k7	1929.14	1978.60	16.89	E-n76-k8	5025.22	5289.70	55.07
A-n53-k7	2265.61	2323.70	26.61	E-n76-k14	5881.17	6190.70	26.09
A-n54-k7	2434.97	2497.40	15.51	E-n101-k14	11,437.05	12,039.00	45.71
A-n55-k9	2701.73	2771.00	24.94	P-n16-k8	711.49	737.30	4.23
A-n60-k9	3261.77	3345.40	17.21	P-n19-k2	351.16	363.90	4.14
A-n61-k9	3271.81	3355.70	31.62	P-n20-k2	340.68	353.04	6.06
A-n62-k8	3279.22	3363.30	21.21	P-n21-k2	385.89	399.88	5.76
A-n63-k9	3559.92	3651.20	19.42	P-n22-k2	398.66	413.22	6.37
A-n63-k10	3705.10	3800.10	19.24	P-n22-k8	543.12	562.84	4.56
A-n64-k9	3736.01	3831.80	21.68	P-n23-k8	593.98	615.53	7.39
A-n65-k9	3757.85	3854.20	45.05	P-n40-k5	1184.54	1227.50	24.96
A-n69-k9	4349.38	4460.90	23.99	P-n45-k5	1514.47	1569.40	16.66
A-n80-k10	6331.26	6493.60	27.33	P-n50-k7	1954.70	2025.60	25.17
B-n31-k5	815.78	828.20	8.93	P-n50-k8	2067.32	2078.03	21.25
B-n34-k5	894.88	908.51	8.13	P-n50-k10	2265.63	2347.80	25.50
B-n35-k5	953.95	998.94	10.74	P-n51-k10	2363.38	2449.10	26.02
B-n38-k6	1201.60	1219.90	16.31	P-n55-k7	2444.25	2532.90	37.64
B-n39-k5	1222.39	1241.00	8.98	P-n55-k8	2556.14	2648.85	40.15
B-n41-k6	1371.22	1392.10	20.29	P-n55-k10	2775.92	2876.60	23.44
B-n43-k6	1479.47	1502.00	17.12	P-n55-k15	2946.63	3053.50	25.09
B-n44-k7	1598.95	1623.30	15.29	P-n60-k10	3273.96	3392.70	35.03
B-n45-k5	1597.37	1621.70	10.46	P-n60-k15	3857.30	3997.20	22.46
B-n45-k7	1623.83	1657.70	10.76	P-n65-k10	3747.10	3883.00	24.35
B-n50-k7	2142.18	2174.80	13.03	P-n70-k10	4478.18	4640.60	46.41
B-n50-k8	2136.66	2169.20	21.77	P-n76-k4	4877.79	5054.70	56.43
B-n51-k8	2195.07	2228.50	26.16	P-n76-k5	4767.10	4940.00	68.13
B-n52-k8	2254.96	2289.30	15.25				

To sum up, as shown in Table 16, it can be concluded that ICS can result in a better balance between solution quality and algorithm efficiency than all the other algorithms.

Table 16. Summary of the performance evaluation.

Problem Set	Algorithm										
	UFC		LNSi			LNS-ACO			ICS		
%Dev( )	Worst	Best	Worst	Best	Time(s)	Worst	Best	Time(s)	Worst	Best	Time(s)
A	6.64	3.57	3.00	2.05	2277.27	1.52	0.84	2288.95	3.50	1.16	18.99
B	5.73	3.32	2.83	1.64	2452.25	0.55	0.35	2489.59	2.83	1.15	21.03
E	6.63	4.34	3.59	2.16	3688.75	1.33	0.68	3882.90	2.89	1.60	24.81
P	7.26	4.66	3.38	1.97	2191.28	0.85	0.48	2270.75	2.62	1.48	24.23

7. Conclusions and Perspectives

This study aimed at developing a novel improved Cuckoo Search (ICS) algorithm by introducing a new cuckoo category which is more intelligent for reducing transport emissions in patient transportation. The capacitated vehicle routing problem (CVRP) is implemented in the research problem and numerical experiments were executed on CVRP instances to validate the effectiveness of the proposed ICS.

Main idea of the new cuckoo category was to enhance the search capability of the Cuckoo Search for the vehicle routing problem. Moreover, two heuristic methods (savings method and randomize

nearest neighbour heuristic) were embedded in host nest initialization and the nearest neighbour heuristic was merged for generating part of the new individuals. The comparison between CS-Ouaarab and ICS in terms of solution quality proved that the new cuckoo category and the heuristic solution construction mechanism brought better intensification to CS. Besides, the performance of the improved CS (ICS) has been tested on a set of classical instances against three recently published methods that deal with CVRP: UFC, LNSi and LNS-ACO. Computational results indicate that the improved CS outperforms both UFC and LNSi in terms of not only solution quality but also algorithm efficiency. As for LNS-ACO, the experimental results show that although LNS-ACO can obtain better solutions than ICS in general case but the gaps are not significant. In some instances, ICS can result in better solutions than LNS-ACO as well. Furthermore, the ICS is much more efficiency than both LNSi and LNS-ACO for all the testing instances.

To sum up, it can be concluded that the management of intensification and diversification through Lévy flights and the new cuckoo category can greatly improve the intensification of algorithm to obtain a better balance between solution quality and algorithm efficiency. The results are quite encouraging.

The next following work can be extended in two different directions. First of all, we will concentrate on the patient transportation problem with transport emissions of real traffic condition. It is known that such emission varies under different traffic conditions, in consequence the uncertainty of real traffic condition makes reducing emissions becoming an interesting and challenging task. Also, future researches will focus on implementing the proposed improved CS algorithm to different combinatorial optimization problems such as different types of vehicle routing problems.

Acknowledgments: This study is partly funded by Shanghai PuJiang Program 13PJC061. Also, the first author thanks the China Scholarship Council for financial support gratefully (contract No. 201504490059).

Author Contributions: Mahjoub Dridi, Amir Hajjam-El-Hassani and Wanlong Lin conceptualized the study. Liyang Xiao developed the algorithm and drafted the manuscript. Hongying FEI designed the experiments and revised the manuscript.

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

References

1. Gulliver, J.; Morley, D.; Vienneau, D.; Fabbri, F.; Bell, M.; Goodman, P.; Beevers, S.; Dajnak, D.; Kelly, F.J.; Fecht, D. Development of an open-source road traffic noise model for exposure assessment. *Environ. Model. Softw.* **2015**, *74*, 183–193.
2. Environmental Protection Agency. *Inventory of US Greenhouse Gas Emissions and Sinks: 1990–2015*; Environmental Protection Agency: Washington, DC, USA, 2017.
3. Toth, P.; Vigo, D. *Vehicle Routing: Problems, Methods, and Applications*; SIAM: Philadelphia, PA, USA, 2014.
4. Golden, B.L.; Raghavan, S.; Wasil, E.A. *The Vehicle Routing Problem: Latest Advances and New Challenges*; Springer Science & Business Media: Berlin, Germany, 2008; Volume 43.
5. Kuznietsov, K.A.; Gromov, V.A.; Skorohod, V.A. Cluster-based supply chain logistics: A case study of a Ukrainian food distributor. *IMA J. Manag. Math.* **2017**, *28*, 553–578.
6. Zhen, L.; Sheng, S.; Xie, Z.; Wang, K. Decision rules for ambulance scheduling decision support systems. *Appl. Soft Comput.* **2015**, *26*, 350–356.
7. Zhou, Y.; Lee, G.M. A Lagrangian Relaxation-Based Solution Method for a Green Vehicle Routing Problem to Minimize Greenhouse Gas Emissions. *Sustainability* **2017**, *9*, 776.
8. Zhang, Z.; Liu, M.; Lim, A. A memetic algorithm for the patient transportation problem. *Omega* **2015**, *54*, 60–71.
9. Ouaarab, A.; Ahiod, B.; Yang, X.S. Discrete cuckoo search algorithm for the travelling salesman problem. *Neural Comput. Appl.* **2014**, *24*, 1659–1669.
10. Laporte, G.; Gendreau, M.; Potvin, J.Y.; Semet, F. Classical and modern heuristics for the vehicle routing problem. *Int. Trans. Oper. Res.* **2000**, *7*, 285–300.
11. Zhen, L.; Xu, Z.; Wang, K.; Ding, Y. Multi-period yard template planning in container terminals. *Transp. Res. Part B* **2016**, *93*, 700–719.

12. Zhen, L.; Yu, S.; Wang, S.; Sun, Z. Scheduling quay cranes and yard trucks for unloading operations in container ports. *Ann. Oper. Res.* **2016**, 1–24, doi:10.1007/s10479-016-2335-9.
13. Gendreau, M.; Hertz, A.; Laporte, G. A tabu search heuristic for the vehicle routing problem. *Manag. Sci.* **1994**, *40*, 1276–1290.
14. Toth, P.; Vigo, D. The granular tabu search and its application to the vehicle-routing problem. *Inf. J. Comput.* **2003**, *15*, 333–346.
15. Osman, I.H. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Ann. Oper. Res.* **1993**, *41*, 421–451.
16. Baker, B.M.; Ayechew, M. A genetic algorithm for the vehicle routing problem. *Comput. Oper. Res.* **2003**, *30*, 787–800.
17. Prins, C. A simple and effective evolutionary algorithm for the vehicle routing problem. *Comput. Oper. Res.* **2004**, *31*, 1985–2002.
18. Shi, Y.; Boudouh, T.; Grunder, O. A hybrid genetic algorithm for a home health care routing problem with time window and fuzzy demand. *Expert Syst. Appl.* **2017**, *72*, 160–176.
19. Zhen, L. Tactical berth allocation under uncertainty. *Eur. J. Oper. Res.* **2015**, *247*, 928–944.
20. Zhen, L. Modeling of yard congestion and optimization of yard template in container ports. *Transp. Res. Part B* **2016**, *90*, 80–104.
21. Cordeau, J.F.; Gendreau, M.; Laporte, G.; Potvin, J.Y.; Semet, F. A guide to vehicle routing heuristics. *J. Oper. Res. Soc.* **2002**, *53*, 512–522.
22. Cordeau, J.F.; Laporte, G.; Savelsbergh, M.W.; Vigo, D. Vehicle routing. *Handb. Oper. Res. Manag. Sci.* **2007**, *14*, 367–428.
23. Gendreau, M.; Potvin, J.Y.; Bräumlaysy, O.; Hasle, G.; Løkketangen, A. Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography. In *The Vehicle Routing Problem: Latest Advances and New Challenges*; Springer: Berlin, Germany, 2008; pp. 143–169.
24. Yesodha, R.; Amudha, T. A study on bio-inspired metaheuristics for solving vehicle routing problem. *Indian J. Sci. Technol.* **2015**, *8*, 1.
25. Thangiah, S.R.; Nygard, K.E.; Juell, P.L. Gideon: A genetic algorithm system for vehicle routing with time windows. In Proceedings of the Seventh IEEE Conference onArtificial Intelligence Applications, Miami Beach, FL, USA, 24–28 Feburary 1991; pp. 322–328.
26. Pereira, F.B.; Tavares, J. *Bio-Inspired Algorithms for the Vehicle Routing Problem*; Springer: Berlin, Germany, 2008; Volume 161.
27. Yu, B.; Yang, Z.Z. An ant colony optimization model: The period vehicle routing problem with time windows. *Transp. Res. Part E* **2011**, *47*, 166–181.
28. Zhou, Y.; Luo, Q.; Xie, J.; Zheng, H. A Hybrid Bat Algorithm with Path Relinking for the Capacitated Vehicle Routing Problem. In *Metaheuristics and Optimization in Civil Engineering*; Springer: Berlin, Germany, 2016; pp. 255–276.
29. Osaba, E.; Carballedo, R.; Yang, X.S.; Diaz, F. An Evolutionary Discrete Firefly Algorithm with Novel Operators for Solving the Vehicle Routing Problem with Time Windows. In *Nature-Inspired Computation in Engineering*; Springer: Berlin, Germany, 2016; pp. 21–41.
30. Tan, L.; Lin, F.; Wang, H. Adaptive comprehensive learning bacterial foraging optimization and its application on vehicle routing problem with time windows. *Neurocomputing* **2015**, *151*, 1208–1215.
31. Yang, X.S.; Deb, S. Cuckoo search via Lévy flights. In Proceedings of the NaBIC 2009 World Congress on Nature & Biologically Inspired Computing, Coimbatore, India, 9–11 December 2009; pp. 210–214.
32. Yang, X.S.; Deb, S. Engineering optimisation by cuckoo search. *Int. J. Math. Model. Numer. Optim.* **2010**, *1*, 330–343.
33. Yildiz, A.R. Cuckoo search algorithm for the selection of optimal machining parameters in milling operations. *Int. J. Adv. Manuf. Technol.* **2013**, *64*, 55–61.
34. Smit, R.; Ntziachristos, L.; Boulter, P. Validation of road vehicle and traffic emission models—A review and meta-analysis. *Atmos. Environ.* **2010**, *44*, 2943–2953.
35. Hausberger, S.; Rexeis, M.; Zallinger, M.; Luz, R. *Emission Factors from the Model PHEM for the HBEFA Version 3*; Report Nr. I-20/2009 Haus-Em; Graz University Technology: Graz, Austria, 2009; Volume 33, p. 679.

36. Knörr, W.; Heidt, C.; Schacht, A. *Aktualisierung? Daten-und Rechenmodell: Energieverbrauch und Schadstoffemissionen des Motorisierten Verkehrs in Deutschland 1960–2030?(TREMOD, Version 5.3) für die Emissionsberichtserstattung 2013 (Berichtsperiode 1990–2011)*; Endbericht, Ifeu Institut: Heidelberg, Germany, 2012.
37. Shang, J.; Zheng, Y.; Tong, W.; Chang, E.; Yu, Y. Inferring gas consumption and pollution emission of vehicles throughout a city. In Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 24–27 August 2014; pp. 1027–1036.
38. Augerat, P.; Belenguer, J.; Benavent, E.; Corberán, A.; Naddef, D.; Rinaldi, G. Computational Results with a Branch and Cut Code for the Capacitated Vehicle Routing Problem; *Rapport de Recherche- IMAG*; Istituto di Analisi dei Sistemi ed Informatica, CNR: ROMA, Italy, 1995.
39. Payne, R.B.; Sorensen, M.D. *The Cuckoos*; Oxford University Press: Oxford, UK, 2005.
40. Croes, G.A. A method for solving traveling-salesman problems. *Oper. Res.* **1958**, *6*, 791–812.
41. Martin, O.; Otto, S.W.; Felten, E.W. Large-step Markov chains for the traveling salesman problem. *Complex Syst.* **1991**, *5*, 299–326.
42. Bräysy, O.; Gendreau, M. Vehicle routing problem with time windows, Part I: Route construction and local search algorithms. *Transp. Sci.* **2005**, *39*, 104–118.
43. Beasley, J.E. Route first—Cluster second methods for vehicle routing. *Omega* **1983**, *11*, 403–408.
44. Clarke, G.; Wright, J.W. Scheduling of vehicles from a central depot to a number of delivery points. *Oper. Res.* **1964**, *12*, 568–581.
45. Prins, C. A GRASP× evolutionary local search hybrid for the vehicle routing problem. In *Bio-Inspired Algorithms for the Vehicle Routing Problem*; Springer: Berlin, Germany, 2009; pp. 35–53.
46. Feo, T.A.; Bard, J.F. Flight scheduling and maintenance base planning. *Manag. Sci.* **1989**, *35*, 1415–1432.
47. Christofides, N.; Eilon, S. An algorithm for the vehicle-dispatching problem. *J. Oper. Res. Soc.* **1969**, *20*, 309–318.
48. Stanojević, M.; Stanojević, B.; Vujošević, M. Enhanced savings calculation and its applications for solving capacitated vehicle routing problem. *Appl. Math. Comput.* **2013**, *219*, 10302–10312.
49. Mohammed, M.A.; Ghani, M.K.A.; Hamed, R.I.; Mostafa, S.A.; Ibrahim, D.A.; Jameel, H.K.; Alallah, A.H. Solving vehicle routing problem by using improved K-nearest neighbor algorithm for best solution. *J. Comput. Sci.* **2017**, *21*, 232–240.
50. Ewbank, H.; Wanke, P.; Hadi-Vencheh, A. An unsupervised fuzzy clustering approach to the capacitated vehicle routing problem. *Neural Comput. Appl.* **2016**, *27*, 857–867.
51. Akpinar, S. Hybrid large neighbourhood search algorithm for capacitated vehicle routing problem. *Expert Syst. Appl.* **2016**, *61*, 28–38.