symmetry

Article

Smart Root Search (SRS) in Solving Service Time–Cost Optimization in Cloud Computing Service Composition (STCOCCSC) Problems

Narjes Khatoon Naseri [1,*], Elankovan Sundararajan [1] and Masri Ayob [2]

1 Centre of Software Technology and Management, Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi 43600, Selangor, Malaysia
2 Data Mining and Optimization Research Group (DMO), Centre for Artificial Intelligent (CAIT), Universiti Kebangsaan Malaysia, Bangi 43600, Selangor, Malaysia
* Correspondence: narges.naseri85@gmail.com

Abstract: In this paper, the novel heuristic search algorithm called Smart Root Search (SRS) was examined for solving a set of different-sized service time–cost optimization in cloud computing service composition (STCOCCSC) problems, and its performance was compared with those of the ICACRO-C, ICACRO-I, ICA, and Niching PSO algorithms. STCOCCSC is an np-hard problem due to the large number of unique services available as well as the many service providers who provide services with different quality levels. Finding closer-to-optimal solutions supports cloud clients by providing them with higher quality-lower price services. The SRS obtained results proved that the SRS provided 6.74, 11.2, 47.95, and 87.29 percent performance improvement on average to the comparative algorithms, respectively, for all considered five problems. Furthermore, employing symmetry concepts in dividing the problem search space helps the algorithm to avoid premature convergence and any efficiency reduction while facing higher-dimensional search spaces. Due to these achievements, the SRS is a multi-purpose, flexible, and scalable heuristic search algorithm capable of being utilized in various optimization applications.

Keywords: combinatorial optimization problem; NP-hard problem; heuristics method; nature-inspired algorithm; cloud computing; quality of service; service time–cost; service composition

check for updates

Citation: Naseri, N.K.; Sundararajan, E.; Ayob, M. Smart Root Search (SRS) in Solving Service Time–Cost Optimization in Cloud Computing Service Composition (STCOCCSC) Problems. *Symmetry* **2023**, *15*, 272. https://doi.org/10.3390/sym15020272

Academic Editor: José Carlos R. Alcantud

Received: 14 December 2022
Revised: 21 December 2022
Accepted: 24 December 2022
Published: 18 January 2023

1. Introduction

Non-deterministic polynomial-time hard (NP-hard) problems are a set of problems for which there is no nondeterministic Turing machine able to solve them in polynomial time. In recent years, a wide range of intelligent phenomena of nature has been considered the source for proposing several search algorithms proposed to solve NP-hard problems. Different aspects of everyday habits and collective wisdom of animals, physical laws of dynamics and electromagnetism, and sociopolitical behaviors of countries have been used repeatedly. It is critical to note that although each of the intelligent search algorithms exhibits its own set of efficiencies and is capable of solving a variety of optimization problems, some common issues exist between them when it comes to solving large-scale, high-dimensional search problems. As part of these issues, the dangers of falling into the trap of local optimizations and the absence of features necessary for endogenous exploitation are two that seem impossible to overcome [1]. As a result, a lot of researchers are trying to come up with novel methods to deal with these problems. Especially, these raised problems in the previously designed nature-inspired stochastic algorithms, have motivated the researcher to look for a new type of intelligent search algorithm that is capable of resolving the existing problems with the present stochastic search algorithms. Accordingly, proposing a new intelligent search algorithm inspired by another type of natural intelligence aimed at dealing effectively with the aforementioned challenges have attracted attention.

Aimed at providing solutions to the aforementioned problems that appeared in the classic search algorithms, stochastic search algorithms have been designed, which are known to be intelligent and efficient [2]. The search process in stochastic search algorithms, not being a linear approach, employs different kinds of random and intelligent selections and searches in order to find the near-to-optimal solution in an acceptable time in the cases that high-dimensional search spaces are being encountered [3]. Stochastic search algorithms are mostly of nature-inspired types; however, some non-nature-inspired algorithms also exist in this category. The well-known nature-inspired algorithms are the Genetic Algorithms (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Artificial Fish Swarm Algorithm (AFSA), Gravitational Attraction Search (GAS), Artificial Immune System (AIS), and Artificial Bee Colony (ABC) [4–6]. In addition, the so-called Imperialist Competitive Algorithm (ICA) [7] and Differential Evolutionary Algorithm (DE) [8] belong to the non-nature-inspired class of stochastic algorithms.

Motivated by the point that although plants are considered intelligent creatures that demonstrate their intelligence in shoot and root development, no strong research studies had focused on utilizing the natural intelligence of plants' roots as a search algorithm, the new, independent, growth-inspired Smart Root Search (SRS) algorithm was proposed in [9,10] to achieve high efficiency and overcome the abovementioned problems.

As the result of another research, the authors proposed a novel search algorithm called SRS in [9,10]. In this study, to evaluate and prove the performance of the SRS, it is applied to solve the real-world problem of the Cloud Computing Service Composition (CCSC) [11] type problems, namely the service time–cost optimization problem in CCSC (STCOCCSC). The obtained experimental results by the SRS are presented, analyzed, compared with those provided in the existing literature on STCOCCSC solutions, and statistically tested. Then, a complete discussion of the results is provided.

The main motivation of the researchers in selecting STCOCCSC is the high dimensionality of the search space as well as complexity of taking different quality of service (*QoS*) [12,13] and experience (*QoE*) [14] parameters, making the optimal solution very difficult to find [11,15].

By taking advantages of the symmetry concepts, the SRS is capable of tackling high-dimensional search spaces and directing the search mechanism towards focusing on the best areas of search space. This approach avoids exploring the non-promising areas and as a result, makes the optimal solution easier to reach.

The remainder of the paper is organized as follows. After a preliminary discussion on the SRS and our motivation to consider the STCOCCSC problems provided in Section 1.1, the STCOCCSC problems and the SRS algorithm are described in Section 2. Section 3 characterizes the conducting experiments to evaluate the SRS and comparative algorithms ICACRO-C, ICACRO-I, ICA, and Niching PSO, including the explanations of the selected problems, experimental design, employed dataset details, and SRS specialization to be applied in solving the STCOCCSC problem. Then, in Section 4, the obtained experimental results by the SRS are presented and analyzed, compared with those provided in the existing literature on STCOCCSC solutions, and statistically tested. Finally, the conclusion and suggestions for future work are presented in Section 5.

1.1. Literature Review

The point of interest in the SRS, proposed by [9,10], was that unlike the previous plant root-based search algorithms, the proposed search algorithm had to be in absolute alignment with optimization principles and be able to overcome the drawbacks that influence other heuristic search algorithms, i.e., the local optima problem, premature convergence, and weak exploitation ability. The SRS provided unique and well-defined features and operators that were precisely extracted from plant intelligence, imitating plant root intelligent foraging behaviors. The smart functionality of the SRS model was based on different intelligent functions of different parts of plant roots in analyzing the heterogeneous conditions of different soil sections. This model divided the search space into subspaces to focus on promising regions. Employing mature, immature, and hair roots allowed the SRS to exploit and explore simultaneously based on the opportunities provided by various parts of the search space. A mechanism involving branching and drouth operations also stimulated the SRS to focus its searching efforts on more promising areas while ignoring unpromising sections in order to search the relatively viable parts and become stronger. To examine the performance and evaluate the efficiency of the SRS in solving classic test functions and real-world optimization problems, two experimental tests were conducted in [9,10]. Several unimodal and multimodal test functions were used to evaluate the SRS. We compared the results with those obtained using GA, PSO, DE, and ICA and analyzed them statistically. Based on the tests that were conducted on the unimodal and multimodal test functions, the SRS outperformed similar algorithms by 91.67% and 81.81%, respectively. Generally, after developing a solution method to solve a particular problem, the proposed method is required to be proved mathematically and then, as a proof, demonstrated using computer programs even though, for many real-world applications, proving a given methodology mathematically is not straightforward, if it is possible at all. In these cases, empiricism proof is used [16]. Classified in [17], this research method in computer science includes three distinct methodologies, namely the theoretical, experimental, and simulation methods. Because of the complexity of the real-world problems caused by factors, including the real-time requests, high multi-modality, non-differentiable nature of the objective function, large, and sometimes deeply constrained search spaces which bring up complexity even in the modeling phase [18,19], the experimental proof has been widely applied. In this context, this paper focuses on applying the proposed SRS algorithm [9,10] to solve the well-known real-world problem STCOCCSC.

In cloud computing and sensor networks [20,21], there are two main challenges to be considered that specifically concern the significance of the accessibilities of all needed services and the efficient allocation possibilities [22]. In the first challenge, predicting all the possibly needed services requires software services, which is an extremely difficult procedure. Simple and single fundamental services are required to be designed and provided by different service providers to be employed in facing this problem. On the other hand, the second challenge usually arises during the selection process of the optimum required single services supplied by various service providers that possess variant quality of service (*QoS*) attributes. Investigating this challenge as an optimization problem can be considered an NP-hard problem [23] since it is subject to a huge number of analogous single services provided by several services in the cloud [24].

Dating back to 2000, service composition was primarily introduced for web services [25–28]. Service composition techniques attracted attention for use in cloud computing systems first in 2009 [29,30], which was followed by substantial efforts, increasing interest, and extensive application of this method in different areas. The variety of novel frameworks, mechanisms, approaches, and algorithms in addition to expanding the scope of problems faced by researchers in this promising area brought up the necessity of recognizing different existing datasets and efficient *QoS* parameters. There are over 20 *QoS* parameters, such as time, cost, privacy [31], and network quality, involved in CCSC [11,32] that must be taken into consideration while dealing with CCSC. Various implementation environments with dependability and resilient [33] complications are other factors playing important roles.

2. Problem and Algorithm Description

In this section the STCOCCSC problem is described followed by discussing the objective function of the problem. Then, the main elements of the SRS algorithm will are introduced, and the required references are given for further details.

2.1. Service Time–Cost Optimization in Cloud Computing Service Composition (STCOCCSC)

Development in the procedures of the computer-based system has resulted in a growing complexity in the services requiring and executing processes. This complexity and the variety of systems make it difficult for an independent service to be sufficiently capable of satisfying the functional prerequisites of the various real-world requests. This propounds the necessity of preparing the simple atomic services as a set, whose components must work together effectively in order to achieve a complex service. Thus, a cloud service composer system (CSC) is required to be combined with the cloud computing.

Accompanied by the growing number of service providers, the similarities can potentially occur within unique services located at different parts of the network. Therefore, for each requirement, the CSC should make efficient decisions to select a unique service amongst the analogous services provided by different service providers. Then, the most proper selected method will lead to the maximum QoS according to the requirements and priorities of customers.

Now, noting that the composite service (CS) in a cloud is assumed to consist of n USs, the QoS parameters of each USs are considered to be the associated service cost (SC) and the service time (ST). A combination of unique services is then required to act correspondingly in an ordinal workflow (wf) in order to arrange for the desired CS. Considering wf_k as the workflow of CS_k, the $SC(wf_k)$ and $ST(wf_k)$ can be defined as the SC and ST of the corresponding workflow k, while the ST and SC vectors of the workflow are described by Equations (1) and (2), respectively [11,15].

$$ST(wf_k) = (ST_1(wf_k), ST_2(wf_k), \ldots, ST_n(wf_k)), \tag{1}$$

$$SC(wf_k) = (SC_1(wf_k), SC_2(wf_k), \ldots, SC_n(wf_k)). \tag{2}$$

The merit value of wf_k, which is defined as the sum of the total service times and total service costs of all elements of wf_k, can be obtained using Equation (3) in which CW and TW indicate, respectively, the weights of cost and time that are assumed to be determined by the user in the range of [0, 1], in such a way that $TW + CW = 1$. In order to define the optimal solution for the STCOCCSC problem, the solution must possess a minimum merit value so that it can be considered the best solution [11,15].

$$MV(wf_k) = \sum_{i=1}^{n} (TW \times ST_i(wf_k) + CW \times SC_i(wf_k)) \tag{3}$$

It should be emphasized that service time and cost values used in the above equations are required to be normalized in the range of 1 to 10 via min–max normalization [34].

2.2. Smart Root Search (SRS) Algorithm

Aimed at overcoming the weakness and shortcomings of the other search algorithms by equipping the SRS with the abilities to simultaneously explore and exploit the search space and circumvent the trap of local optima caused by premature convergence, the proposed SRS algorithm [9,10] was designed to possess some distinguished characteristics, taking into account the plant root intelligence. One of the basic steps in designing the SRS was providing a map between the optimization terms and real plant root growth terms. In this context, the following mappings were considered:

- The soil environment for plant roots was interpreted as the search space which contains all the problem's possible candidate solutions.
- The plant root set was considered as the solutions' vector.
- The root was regarded as the solution.
- The nitrate concentration was considered as the objective function employed for evaluating the solution.
- The location of the highest nitrate concentration was considered as the optimal solution, whose objective function value is minimized.
- The growth steps were interpreted as iterations.
- Hair roots germination represented the local search operator.
- Root growth was interpreted as the solution movement.

- The concept of root drouth represented solution elimination.
- The root growth speed was interpreted as the velocity of movement of the solution.
- The concept of root branching was interpreted as solution reproduction.
- The immature root was considered as a limited-move solution.
- The growth direction was regarded as a movement coefficient set.

In the following, we describe the main procedures of the SRS algorithm, which include parameter initialization, search space division, initialization of the first generation, the roots' evaluation, sorting and ranking, growth of roots, roots drouth, and branching, in addition to HRG and termination criteria, which are illustrated in the SRS flowchart in Figure 1.

To adjust the SRS search elements and operations, setting a few guiding parameters is one of the first steps. The values taken by these parameters are affected by the problem specifications. The parameters will be gradually defined within the following paragraphs.

In the expansive search spaces, the search algorithms deal with a massive number of space points that require probing. Generally, the total space point's number in these cases is notably more than the initial solutions number, obtained in the first iteration of the algorithm running. Additionally, since the algorithm's initial solutions are generated randomly, there is not usually a uniform distribution of solutions within the search space. Hence, the search space cannot be exhaustively searched well. In these extensive search spaces, dividing the search space helps algorithms solve the problem more conveniently. In designing the SRS algorithm, a divide-and-conquer strategy [35] was employed to divide the problem search space into Ns number of subspaces. Noting that the convergence speed of the SRS is directly affected by the number of subspaces, an effective Ns initialized value is subject to the problem specifications which are the solutions' structure, sizes of the different dimensions of the search space, and number of dimensions.

The first generations of the solutions, which are randomly generated by the SRS, are assumed to include $NumMinRoot$ number of solutions that are distributed in the search space so that the numbers of initially generated solutions assigned to every subspace are equal. However, in the cases that some solutions remained unassigned, they will be assigned to the subspaces at random. Noting that in SRS terminology any SRS-generated solution is interpreted as a root, for the root i in a D-dimensional problem search space, the location is given by Equation (4), in which x_i^d indicates the location of root i in the specific dimension d.

$$x_i = \left(x_i^1, x_i^2, \ldots, x_i^d, \ldots, x_i^D\right) \tag{4}$$

In the SRS, a root is identified by its structures that are needed to describe the root. The fundamental structure of a root is illustrated in Figure 2. In the first row, the structure of interest is listed, while in the second row, an element of the structure is identified. Once the step of initialization of the first generation ends, all subspaces have an equal number of generated roots.

To provide a procedure for evaluating roots, the detailed effects of the two most effective soil nutrients, phosphate and nitrate, on root growth were provided in [9,10]. To be more specific, root growth speed, hair roots density, and branch density are highly affected by the concentration of these nutrients [6,36–42]. When the phosphate concentration becomes deficient, the primary and LR growth are dramatically modified, causing significant changes in the overall root architecture. In contrast, sufficient nitrate accessibility may influence LR elongation. Primary root elongation has been revealed to be influenced by nitrate and phosphate availability in a contrasting way [36]. In Arabidopsis, it has been observed that increasing nitrate supply leads to a reduction in the primary root elongation. On the other hand, the abundance of phosphate availability increases primary root elongation. Forming hair roots is another manifest change in root architecture that can be caused by a deficiency of some nutrients. Especially, deep investigation on Arabidopsis has demonstrated that in response to phosphate deprivation the hair roots grow longer and denser [36,39,43]. Lateral root density is the other conspicuous factor in root architecture. Research has shown that density of LRs remains indifferent to changes in nitrate concentrations but shows a dramatic response to phosphate supply, as it decreases when the phosphate availability increases [37].

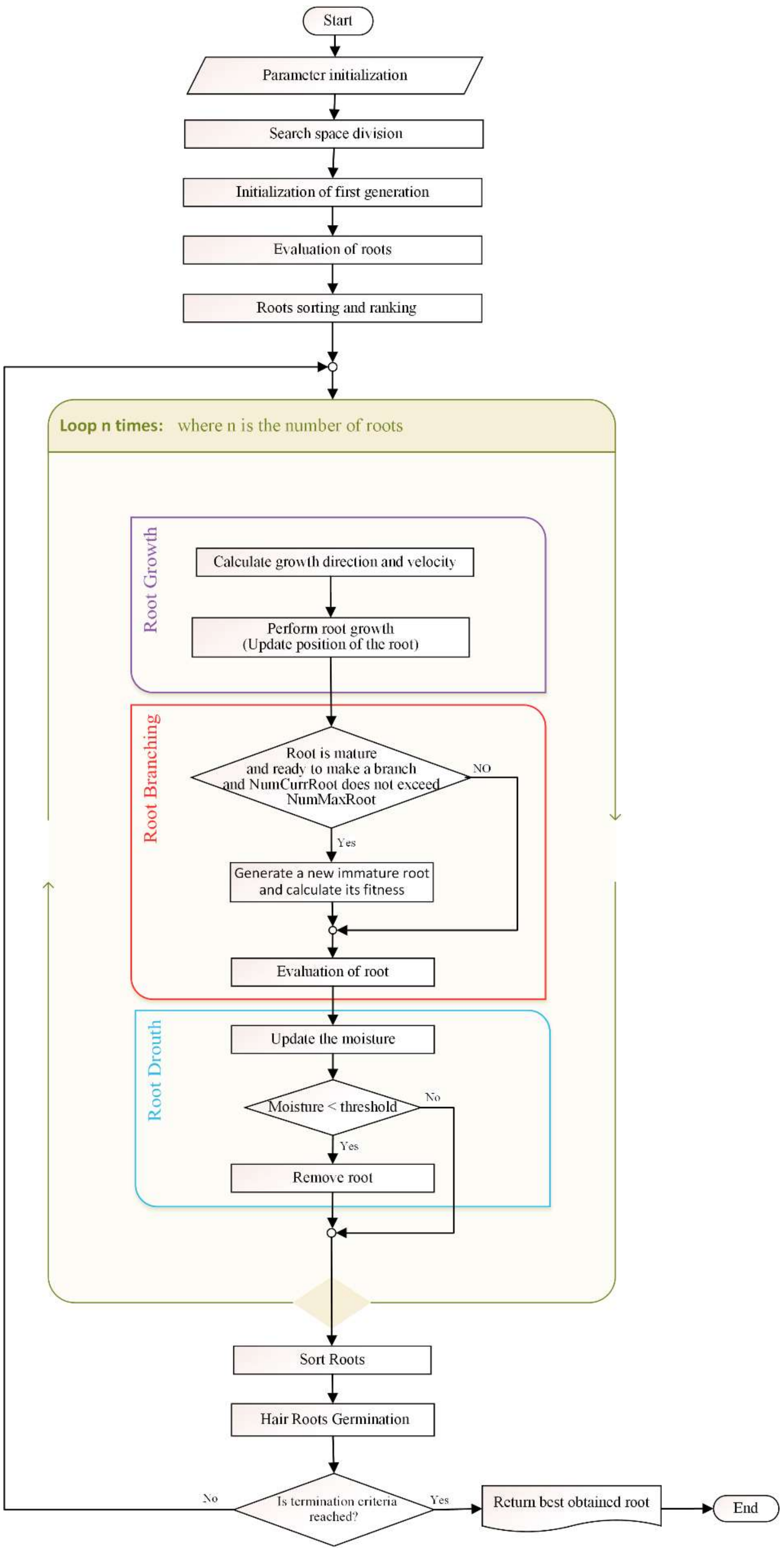

Start
Parameter initialization
Search space division
Initialization of first generation
Evaluation of roots
Roots sorting and ranking
Loop n times: where n is the number of roots
Root Growth
Calculate growth direction and velocity
Perform root growth (Update position of the root)
Root Branching
Root is mature and ready to make a branch and NumCurrRoot does not exceed NumMaxRoot
NO
Yes
Generate a new immature root and calculate its fitness
Evaluation of root
Root Drouth
Update the moisture
Moisture < threshold
No
Yes
Remove root
Sort Roots
Hair Roots Germination
Is termination criteria reached?
No
Yes
Return best obtained root
End

Figure 1. The smart root search (SRS) flowchart.

x^1	x^2	$x^3 \ldots x^{D-1}$	x^D	*Nitrate Concentration*	*Velocity*	*Type*	*Age*
Value$_1$	Value$_2$	Value$_3$ … Value$_{D-1}$	Value$_D$	Object Function Value	Velocity Value	Type of Root	Age Value

Figure 2. An SRS basic root structure.

To apply these behaviors in the SRS, two main points were considered. The SRS is particularly interested in root growth speed. Since the root growth speed can only be considered for the primary type of roots and noting that under different nutrient availabilities the primary root elongation behavior is the same as the primary root growth speed behavior, without loss of generality, the effect on primary root elongation was considered as the effect on the root growth speed. In addition, the production of the LR was considered the task of root branching in the SRS.

Generally, the impact of a high concentration of nitrate on the growth of root speed appeared to be similar to that of the low phosphate concentration. However, when these two nutrients have the same availabilities, i.e., both are high or low, nitrate and phosphate conceal each other's effects, suppressing the primary root growth. The main idea behind combining the several effects of disparate nutrient concentrations was to simplify the proposed model. Then, highlighting the importance of primary root growth in the SRS, the cases in which the high (low) nitrate accompanied by high (low) phosphate supplies, which concluded in almost zero root growth speed, was abandoned. Then, noting the great importance of the reasonable simplicity in proposing the SRS, supported by the evidence, the increasing nitrate concentration was considered to coincide with the decreasing phosphate concentration. This assumption, besides shortening the procedure, facilitated defining the SRS concepts that have more compatibility with the corresponding botany terminology. Then, it suffices to consider only the effects of nitrate as the nutrient that influences root life. Accordingly, the objective function value of each root, indicated by $f(x)$, was considered to be specified by the nitrate concentration of that root, where it was defined by [9,10] as follows:

$$f(x) = Nitrate\ Concentration. \tag{5}$$

Another important mechanism in the SRS is sorting and ranking the roots that are being used for solving the problem based on the objective function values associated with each root. Sorting the existing roots of the SRS is made not only when the initialization step is finished, but also at the end of all execution iterations. In doing so, the ranking of each root becomes manifest among all the current roots as well as among the roots of the corresponding subspace. The advantage of this mechanism is that besides recognizing the best roots of every subspace, it gives the rank of each root in the roots list and finally makes it possible for the SRS to capture and determine which root is the best global root and place it at the top of the roots list. The root ranking and sorting affect root growth, branching, drying up, as well as the SRS convergence.

From a lifetime point of view, the SRS categorizes the roots into two main categories of temporary and permanent roots. The permanent roots, subcategorized into immature and mature roots, are introduced according to the age of the roots. On the other hand, the temporary roots fall into a category called Hair Roots (HRs). Considering these three groups of roots, the root growth mechanism applied in the SRS was designed to result in managing the local convergence in all subspaces.

Regarding the process of growing the mature roots, suppose that root i in D-dimensional search space at time t is located at $x_i(t)$, given by Equation (6). Root growth means that in each dimension d, the root cap moves from its current location, given by $x_i^d(t)$, to a new location, given by $x_i^d(t+1)$. Assuming that the root growth velocity $v(t)$ and direction are determined, the second location is related to the first one by Equation (7), where the velocity in d direction, i.e., $v_i^d(t)$, is the projection of the root velocity vector $v_i(t)$ on the d direction's axis.

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t), \tag{6}$$

$$v_i^d(t) = \left\lceil v_i(t) \times cos\theta_i^d \right\rceil. \tag{7}$$

In this case, $cos\ \theta_i^d$ for root i is the cosine of the angle between the direction d and the root movement direction, which is the direction of velocity as well.

On the other hand, noting that the convergence policy of the SRS constrains the root growth direction toward the best roots, the best root set (BRS) was introduced in [9,10]. The BRS for each subspace was defined to be created by the first k number of the best roots of that subspace. This substantial set affects the growth directions in subspaces in such a way that all roots of a subspace grow toward the closest root from that subspace's BRS. To establish this set, k is determined by utilizing the well-proposed method of "Roulette Wheel Selection via Stochastic Acceptance" (RWSSA) [44]. Once the k number of the best roots is picked to create the BRS of every subspace, the best nearest root to root i can be determined to identify the root growth direction of root i. In order to determine the best nearest root to root i, the SRS uses a parameter called $density_{j,i}$. Density is defined for every root j in BRS by Equation (8) in which NC_j stands for the concentration of nitrate for root j, x_j^d and x_i^d are, respectively, the locations of roots j and i in dimension d, and the Euclidean distance between roots i and j is the dominator.

$$density_{j,i} = \frac{NC_j}{\sqrt{\sum_{d=1}^{D} \left(x_j^d - x_i^d\right)^2}}. \tag{8}$$

Then, among the BRS roots, the best root nearest to root i, indicated by $best\_closest_i$, is defined as a root j which maximizes the $density_{j,i}$. In other words, we have

$$best\_closest_i = \left\{ j \middle| \ density_{j,i} is\ MAX \right\}. \tag{9}$$

Accordingly, the growth velocity and direction of a mature root can be defined. To calculate the velocity of a mature root, there is a need for the maximum value of velocity which is not allowed to be exceeded by the velocity of roots. Referred to as v_{max}, this maximum velocity is assumed to be defined by the user. Then, the velocity of the root can be identified as a function of v_{max}, considering the rank of each root between all roots. In doing so, the velocity is required to decrease for lower ranks of roots. This policy provides an effective exploration since the roots located at promising areas strain to grow slowly and explore the area precisely, while the other roots are required to recede quickly from the non-appropriate areas. Then, if glb_rank_i stands for the global rank of root i among all present roots and $NumCurrRoot$ shows the number of roots that currently exist, the velocity of root i is defined by Equation (10).

$$v_i(t) = v_{max} - \left[v_{max} \left(1 - \frac{glb\_rank_i}{NumCurrRoot} \right) \right] \tag{10}$$

The direction of the mature root i is especially dictated by the location of $best\_closest_i$, such that for every dimension, the root grows toward its best closest root. Therefore, a coefficient needs to be defined for the root's current velocity to demonstrate the root growth direction toward the $best\_closest_i$ by using a suitable dimensional velocity. The coefficient was formulated to choose values that prevent the growth of the current root beyond its closest best root. Now, if θ_i^d is the angle of root i growth toward its $x_{best\_closest_i}^d$, i.e., the best closest root in dimension d, the cosines of θ_i^d are given by Equation (11).

$$cos\theta_i^d = \frac{x^d_{best\_closest} - x^d_i}{\sqrt{\sum_{d=1}^{D}\left(x^d_{best\_closest} - x^d_i\right)^2}}. \tag{11}$$

Then, noting that for mature root i currently located at $x_i^d(t)$, the root velocity is given by Equation (10) and the direction toward $best\_closest_i$ indicates the angle θ given by Equation (11), the next position of root, $x_i^d(t+1)$, can be easily determined via Equation (6).

Despite the mature roots, the immature roots, which are newborn, are not allowed to change their direction and velocity of growing or to generate new branches. There are some rules obeyed by the immature roots. Their growth velocity is constant and specified by their parent as they follow the growth behavior of the parent and grow with the same root velocity. In addition, the growth direction of immature roots is initialized randomly and will be constant.

The SRS uses a maturation mechanism by which an immature root can transform into a mature one. Thus, since the type of root can be changed during iterations, the same root can be used by the algorithm to apply various search policies. As shown in the smart root search (SRS) flowchart in Figure 1, each root possesses an attribute called the root age. As soon as a new root is produced, its age is initialized as 0. After each iteration of the algorithm, the age of the root is increased by one. Once a threshold, called $Mature\_Age$, is reached by the age value assigned to an immature root, the status of the immature root will be changed to "mature". $Mature\_Age$ can be either stated between three and five iterations randomly as it is considered in the SRS or defined by the user, noting the adopted exploitation and exploration policies and considering the fact that the higher $Mature\_Age$ values lead to less exploitation and more exploration and vice versa.

In root growth, branching is a mechanism in which new roots are produced by mature roots to enhance the search rate over not yet investigated parts of the soil. The SRS utilizes a $Branching$ operation in the search space to generate new immature roots, which are adjacent to their mature parent root in the early stages of their lifetime. The basic idea behind designing the root branching was providing the roots located in promising locations the opportunity to create more branching compared to the roots in non-promising areas. To manage the number of branches in the SRS, a score-based mechanism was designed in which the mature roots are granted nitrate concentration-based scores in the range of one to four in every iteration, determined based on their fitness.

The sum of collected scores, referred to as $SCSs$, is one of the effective factors in branching, as the mature root will be allowed to create branches after several iterations provided that the obtained SCS meets a predefined threshold value, called Minimum Required Ability (MRA). As soon as the MRA score is reached by a root, a new root will be generated, resetting the SCS value to 0. Then, the mature root begins to re-collect the concentration-based scores, and the cycle continues.

On the other hand, as it is well-understood from botany research, plant roots absorb the water from the moistened parts of soil and store it to use afterward by transferring the reserved water to drier parts of soil [45–47]. The amount of the stored water of the core root determines whether the root is qualified to make new roots or not. Focusing on this process, a mechanism for root drouth was proposed exclusively in the SRS. In fact, the branching procedure may increase the number of active roots in the search space. In these cases, in order to eliminate the improper roots, an operator was introduced to the SRS, called the Root Drouth operator.

Naturally, a newly generated root possesses a certain amount of water. Noting this fact, to implement the Root Drouth operator, all new roots were considered to have a definite moisture volume, called the Moisture Percentage (MP), at their initialization. MP is a variable, and its initial value in the SRS can be set to 50 as a moderate value. For the mature roots, the MP changes as the root grows. To help immature roots sustain themselves during the pre-pubertal development, their MP values are kept fixed.

In the SRS, to stimulate the plant roots' behavior as they absorb and store the water to use in drier areas and dry when encounter a lack of water supply in dry soil, the root encouragement (punishment) value was introduced, referred to as *Encourage_Value* (*Penalty_Value*). To identify which kind of these values should be assigned to a root in the growth process in the SRS, the current and the previous position of the root need to be compared, considering the objective function values. Then, the *MP* value is increased by the *Encourage_Value* if the root is in a better location and is decreased by the *Penalty_Value* if the root is in a worse location. The roots in non-promising locations are usually given a limited chance to continue searching. However, if the *MP* dwindles to meet (pass) the predetermined drouth threshold, i.e., the minimum accepted value of *MP*, root drouth occurs.

In contrast to the permanent type of plant roots that includes mature and immature roots, hair roots are temporary and short-sized roots, which are generated by mature roots in order to take better advantage of the promising areas of soil [6,48]. The survival period of these short-lived roots is usually about 2 to 3 weeks, and then they die off [6,48]. The hair root inspired the built-in exploitation operation, called the HRG operator, which provides additional local search around the neighborhoods of the best-generated solutions. However, not all the good solutions can be given the chance to be enhanced by local search in their neighborhood. The number of the best roots to be considered in this mechanism, the radius of the neighborhood to be locally searched, and the selected dimensions of the roots to be investigated must be identified.

The SRS, being similar to other heuristic search algorithms, will stop executing when it meets at least one of the following criteria: (1) achieving the desired solution or (2) exceeding the maximum number of iterations. After the algorithm has been executed, the best solution is acknowledged as the final result.

3. Experimental Design

A trustworthy comparison between various algorithms for a specific problem requires a fair comparison environment, which is usually provided by a well-designed Benchmark Problems Set (BPS) to be solved by algorithms. In this study, we take advantage of the BPS for STCOCCSC problems provided in [49] based on the extracted dataset presented in [32] by which the proposed SRS algorithm can be compared to the comparative algorithms. The description of the STCOCCSC dataset, experimental STCOCCSC problems, and the SRS execution are provided in the following.

3.1. STCOCCSC Dataset

A reliable and adequate-sized dataset is the first requirement for an experimental test. The real-world cloud computing systems often include a large service pool. The dataset should involve many service providers presenting a great number of different simple services with several values for the *QoS* parameter. Because the STCOCCSC problems have a great extent and magnitude, the dataset cannot be generated blindly and uniformly randomized, since nearly identical solutions will probably be obtained. As mentioned before, in this work the data collection provided in [32] is utilized, where the WSDREAM-DataSet2 [50] was selected to be improved and then utilized.

Although the WSDREAM dataset has many service providers, it lacks some service time variable values. Noting that excluding these incomplete cases from its 339 service providers may lead to a reduction of the dataset size and a loss of power in data analysis [51], imputing the missing values was used in [32] to manage the problem of missing data and complete the dataset. Among the traditional imputation methods proposed to treat missing values, for instance, mean substitution, complete case analysis, and regression imputation, the innovative approach of multiple-imputation is usually regarded as a general solution [52].

In [32], to prepare the dataset, eliminating the services possessing 30% or more missing values, which are 285 services, from the analysis, a new dataset which includes 5540 services was obtained. The multiple-imputation method was then applied, using the iterative Markov chain Mont Carlo method and considering five iterations. In the dataset, the specific number of iterations is actually indicated by the percentage of missing cases [53]. The used service cost values in the experiments were then prepared in such a way that the assumed cost values were randomly generated in order to reach isodiametric service time and service cost arrays. Practically, determining the statistical distribution which fits the columns of the service time array the best, a column of data was randomly generated for each investigated column of the service time array, reaching an identical distribution of generated data and investigated data. In doing so, a highly reliable dataset was prepared since neither missing values nor outlier data were present.

In the final step of preparing the dataset, the values of service time and service cost were normalized in [32] using the min–max normalization method, where the maximum and minimum were considered, respectively, as 10 and 1. This difference between the maximum and minimum values helped the objective function calculations avoid missing minor differences and rounding the results. Consequently, this makes it possible for algorithms to properly decide between close solutions. Table 1 presents a brief description of the specifications of the used dataset.

Table 1. Specifications of the utilized datasets in the experiments.

Dataset Name	*QoS* Parameters	No. of Service Providers	No. of Presented Services
Jula et al.'s dataset	Service time Service cost	339	5540

3.2. Experimental STCOCCSC Problems

In an effort to prevent instances of STCOCCSC problems in [49] based on their extracted dataset, namely the revised WSDREAM-DataSet2 presented in [32], a set including 5 different-sized problems was introduced, called P1, P2, P3, P4, and P5 with problem sizes of 100, 200, 300, 400, and 500, respectively. The problem size 100, for example, stands for 100 simple services that are required to be composed together to provide a favorable complex service. Therefore, the presented set of STCOCCSC instances ranged from easy to difficult problems. In addition, in generating each problem, they randomly selected a set of m different and unique simple services among provided services in the extracted dataset.

These five generated different-sized STCOCCSC problems were probed by ICA, ICACRO-I, ICACRO-C, and Niching PSO in [49]. Here, in order to compare and evaluate the results obtained by the SRS in solving these problems, we consider the results obtained by the four algorithms provided in [49] based on the dataset provided in [32]. Considering their work, the default parameters for Niching PSO and ICA were established based on [54,55] and [7], respectively, while for ICACRO-C and ICACRO-I, the ICA environment was used. The SRS algorithm was applied on these five STCOCCSC problems utilizing the revised WSDREAM-DataSet2 as a real-world dataset generated and provided in [32].

3.3. SRS Execution

The SRS algorithm was implemented in Microsoft Visual Studio C#.NET 2015. For each problem, the size of the problem indicated the number of unique services required to be combined in preparing the requested composite service. According to the description of the STCOCCSC problem, for any required unique service, the composition algorithm is required to select a service between the 339 similar services in such a way that the associated sum of the service time–cost for all selected services is minimized.

To compare the results of the SRS, we consider the four search algorithms, ICA, ICACRO-I, ICACRO-C, and Niching PSO implemented and executed in [49]. Since the mentioned algorithms were correctly executed 40 times on the experimented problems defined based on the previously mentioned data, the SRS is also executed 40 times, independently for each problem, on a PC with 8 GB of RAM and an Intel Core i7-3.40 GHz processor under identical conditions. Recording the results, the average of the executions was calculated, stored, and observed to be employed for further analysis and comparison.

3.4. SRS Performance Evaluation

The main procedures of the SRS described in Section 2.2 were followed. In addition, for the considered STCOCCSC problems, for which the QoS parameters are the service time and service cost values, the description of the basic structure of the roots generated by the SRS is presented in Figure 3. As a reminder, x_i^d indicates the location of root i in dimension d.

	x^1	x^2	$x^3 \ldots x^D$	*Nitrate Concentration*	*Velocity*	*Type*	*Age*
Service Time	st_1 Value	st_2 Value	$st_3 \ldots st_D$ Values	Object Function Value	Velocity Value	Type of Root	Age Value
Service Cost	sc_1 Value	sc_2 Value	$sc_3 \ldots sc_D$ Values				

Figure 3. Basic structures of the roots generated by the SRS.

Furthermore, the SRS parameter setting applied in solving STCOCCSC problems are provided in Table 2. It should be noted that in [49] to study the results of applying ICA, ICACRO-I, ICACRO-C, and Niching PSO on STCOCCSC problems, the initialized number of countries for the first three algorithms was considered to be 500. In the same way, Niching PSO was initiated with 500 particles. Hence, to arrange an equivalent environment for comparing the SRS and the comparative algorithms presented in [49], the SRS's $NumMaxRoot$ was particularly set to 500 to have the same number of solutions at each iteration. The other parameters were set with respect to definitions and the behavior of the parameters suggested by the SRS proposers. According to [9], v_{max} must be set to one-third of maximum domain value of the problem which is 339. Thus, it was set to 113. *Encourage_Value* and *Penalty_Rate* also follow constant values set in [9]. *MRA* and *Max_Penalty* were suggested in [9] to be proportional to *NumMaxRoot* to keep the proportion of the mature and immature roots over the course of running the algorithm. In this research, high MRA value helps the algorithm keep the immature root at a lower level to help the mature ones explore more. As a complementary setup, we also set the *Mature_Age* at a small number to convert the best immature ones to mature to avoid them being suppressed and give them higher chance to explore. High *Max_Penalty* also punishes inappropriate solutions more to let the algorithm focus more on the promising areas.

Table 2. SRS parameters setting.

Ns	***NumMinRoot* (Population Size)**	*NumMaxRoot*	v_{max}	*MRA*	*Max_Penalty*	*Encourage_Value*	*Penalty_Rate*	*Mature_Age*
8	125	500	0.33 × MDV	70	40	2	0.25	5

On the other hand, defining reliable measures is required to evaluate the performance of the algorithms in solving instances of certain optimization problems in addition to comparing the performance and scalability of the SRS against comparative algorithms. The objective function value of the STCOCCSC problem is considered the merit value, and the best merit value was used as a measure to indicate the quality of the best solution. The nature of the investigated problems determines which one of the extremum values of the objective function designates the best solutions. Since the main point of investigating the STCOCCSC problem is to find solutions with higher QoS, the minimum merit valued solutions are of interest since the cost and service time need to be comparatively lower to reach higher *QoS*.

In addition, in [49], for the five different-sized STCOCCSC problems probed by ICA, ICACRO-I, ICACRO-C, and Niching PSO, the check point to compare the results was considered at iteration 1500. Hence, to acquire more accuracy in evaluations and reliability at comparisons, during an execution process the SRS results are compared to those of the four comparative algorithms at iteration 1500. Comparing the results, the best solver between is determined by ranking the algorithms in reaching the minimum merit value at the end point of the run at iteration 1500. The improvement percentages of the solutions obtained by the SRS are also calculated to demonstrate the efficiency of the SRS in solving these real-world problems.

Furthermore, statistical tests were applied to validate the obtained results by the SRS and comparative algorithms to show whether the results and conclusions were supported by the experiment based on the considered STCOCCSC problem. In this regard, two different statistical tests were implemented using IBM SPSS STATISTICS version 22. First, to determine the difference between the mean merit values found via the SRS and the comparative algorithms, a repeated measures analysis of variance [44–46] performed with the Greenhouse–Geyser correction [47]. The results of this analysis show the statistically significant difference, if it exists, between the mean merit values of the considered algorithms. In this stage, the difference between the performances of two algorithms is considered significant when the null hypothesis of the test is rejected at the 95% confidence level with p-value < 0.05. In the case that the null hypothesis is rejected in the repeated measures analysis of variance, which represents the equivalence of the rankings between the algorithms, a post hoc analysis is conducted in order to compare the highest performing algorithm against the other algorithms. The considered post hoc procedure in this part of the research was the Bonferroni corrected [56–58]. The results of this test can be employed in evaluating and pairwise comparing the functionality and performance of the algorithms, noting the size of the problems.

4. Results and Discussion

For the given problem P1, Figure 4 shows the total service time–cost obtained by the SRS in 1500 iterations, compared to the solutions achieved by ICA, ICACRO-I, and ICACRO-C in [49]. Clearly, the solutions obtained by the SRS at the end of 1500 iterations are considerably better than the other four algorithms. Niching PSO seems to be the first one trapped in the local optima, away from the optimal solutions obtained by ICA, ICACRO-I, ICACRO-C, and the SRS. Although the ICACRO-I and ICACRO-C show comparatively closer performance to the SRS, the SRS sustains its dominance to reach better solutions. The fixed point of 1500 iterations was considered to be the checkpoint of performance comparison; however, it is noteworthy that the SRS overtook the other algorithms after almost 300 iterations. The SRS avoided the local optima trap, and the trend of merit value of the best solutions reveals that the SRS has the potential to reach better solutions with a higher number of iterations.

Similarly, for problems P2, P3, P4, and P5, the achieved performance of the SRS, compared to those of ICA, Niching PSO, ICACRO-C, and ICACRO-I [49], are shown in Figures 5–8. The results indicate that the total service time–cost of the best solutions generated by the SRS, ICACRO-C, and ICACRO-I were consistently lower than the service time–cost of the provided solutions by the ICA and Niching PSO. Focusing on the generated execution results of the SRS and the two types of ICACRO, it can be concluded that, despite the fact that the best solutions produced by ICACROs are neighboring the best solutions generated by the SRS, the established superiority by the SRS is sufficient to reach the optimal solutions.

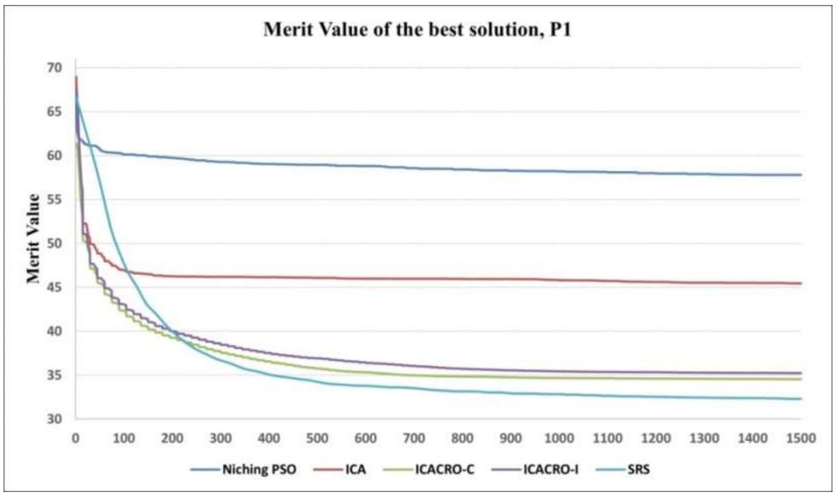


Figure 4. Total service time–cost provided by the five algorithms in 1500 iterations for problem P1.

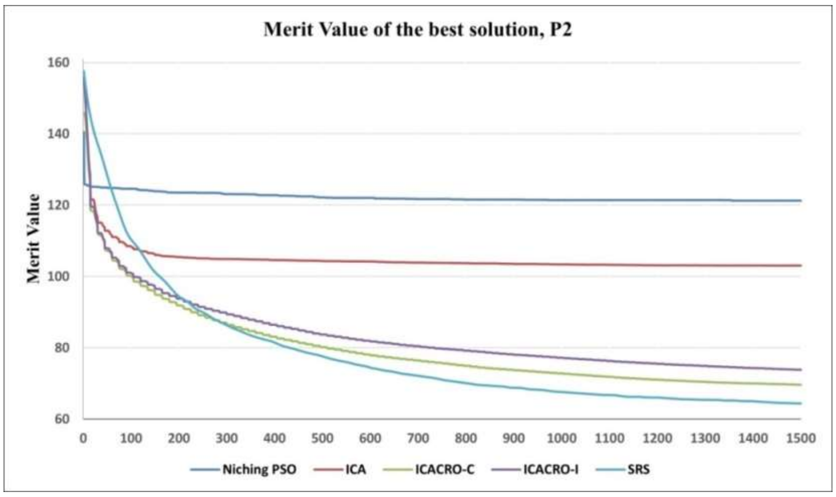


Figure 5. Total service time–cost provided by the five algorithms in 1500 iterations for problem P2.

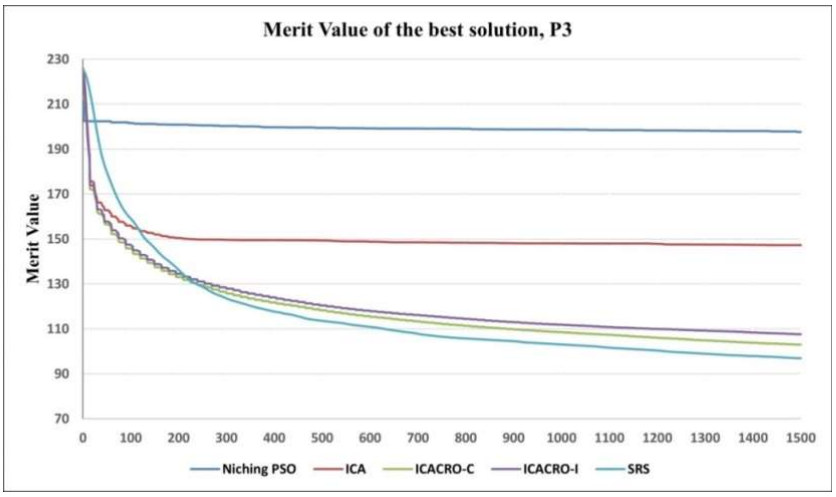


Figure 6. Total service time–cost provided by the five algorithms in 1500 iterations for problem P3.

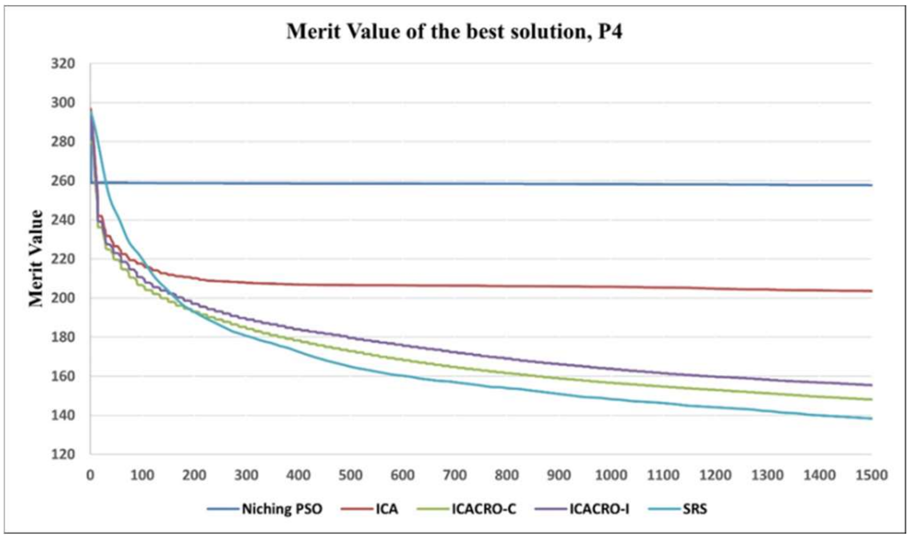


Figure 7. Total service time–cost provided by the five algorithms in 1500 iterations for problem P4.

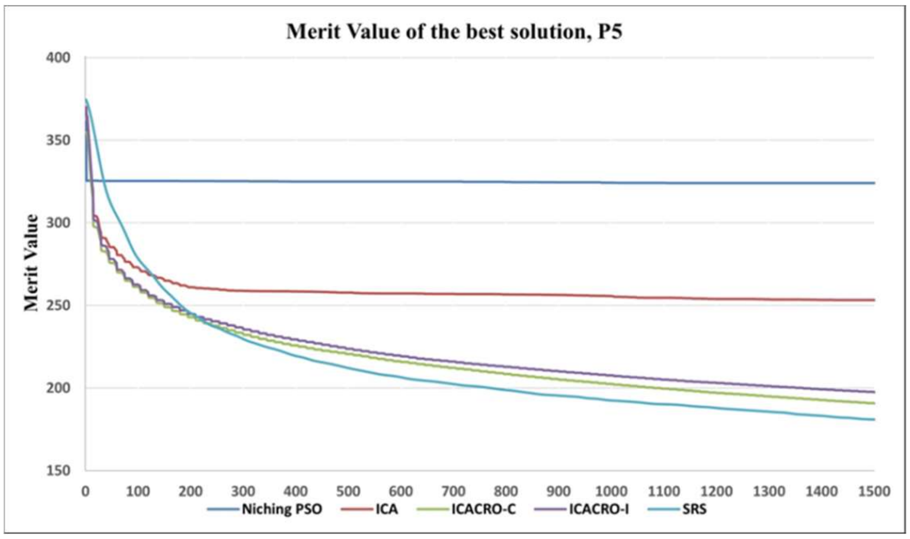


Figure 8. Total service time–cost provided by the five algorithms in 1500 iterations for problem P5.

It is worth mentioning that, noting Figures 4–8, the size of the problems does not alter the competence of the SRS as it overcomes the other algorithms after almost 300 iterations for all problem instances and avoiding the local convergence, obtains the optimal solutions at 1500 iterations. In addition, noting the trends, the SRS predictably reached better solutions with a higher number of iterations. This high performance relies on the efficiency of the SRS and the method in which it is designed.

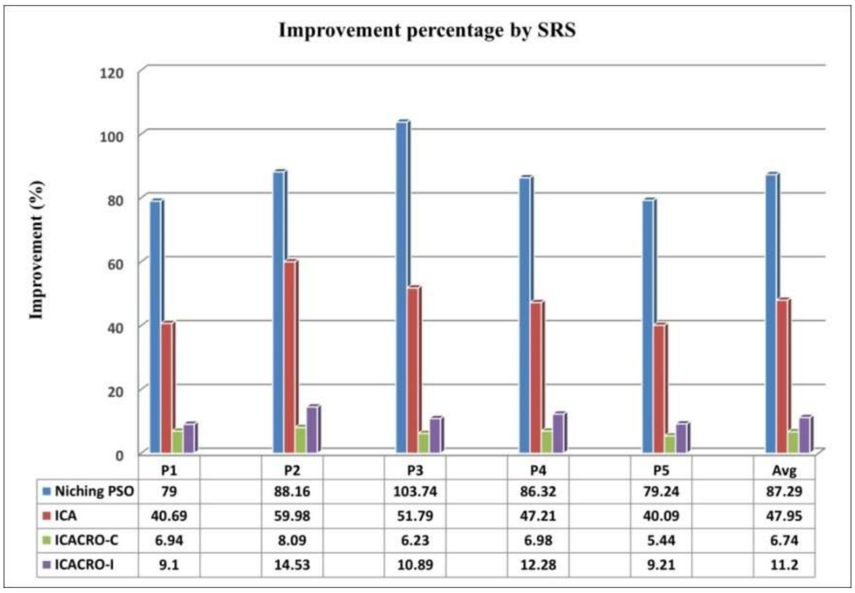
To give another comparative view in solving problems P1 to P5 at the end point of the run at 1500 iterations, the best results obtained by the SRS, along with the provided best results by Niching PSO, ICA, ICACRO-C, and ICACRO-I [49], are summarized and ranked in Table 3. Being indifferent to the size and difficulty of the five problems, the SRS demonstrated the best performance among these five algorithms, producing the ranked first solution. The two types of ICACRO-I and ICACRO-C follow, respectively, the SRS in supplying the best results. Then, after a noteworthy gap, the ICA and Niching PSO presented their best results.

Table 3. Best results after 1500 iterations.

Algorithm		**P1**	**P2**	**P3**	**P4**	**P5**
Niching PSO	Best	57.80	121.21	197.59	257.70	324.07
	Rank	5	5	5	5	5
ICA	Best	45.43	103.06	147.21	203.61	253.28
	Rank	4	4	4	4	4
ICACRO-C	Best	34.53	69.63	103.018	147.96	190.64
	Rank	2	2	2	2	2
ICACRO-I	Best	35.23	73.78	107.54	155.3	197.45
	Rank	3	3	3	3	3
SRS	Best	32.29	64.42	96.98	138.31	180.8
	Rank	1	1	1	1	1

4.1. Improvement Percentage of the Solutions Obtained by the SRS

A numerical analysis of the improvement percentage of the solutions obtained by the SRS with respect to those obtained by Niching PSO, ICA, ICACRO-C, and ICACRO-I in [49] is presented in Figure 9. It includes the improvement percentages for problems P1 to P5, case by case, along with the average improvement percentage. The highest improvements are observed over the Niching PSO solutions, ranging from 79% for P1 to 103.74% for P3. With a notable difference, the second highest improvements are over the ICA results, ranging from 40.09% for P5 to 59.98% for P2. The ICACRO-I solutions are the next promoted solutions, for which the improvement percentage varies from 9.1% for P1 to 12.28% for P4. Finally, having closer results to the SRS, the ICACRO-C gains the least improvement, with 5.44% for P4 up to 8.09% for P2. The maximum and minimum improvement values achieved for problems P1 to P5 are 103.74% for P3 with Niching PSO and 5.44% for P5 with ICACRO-C. No outstanding relation is detected between the improvement percentages and the size of the different problems, demonstrating that the size of search space does not influence the SRS efficiency.



	P1	P2	P3	P4	P5	Avg
🟦 Niching PSO	79	88.16	103.74	86.32	79.24	87.29
🟥 ICA	40.69	59.98	51.79	47.21	40.09	47.95
🟩 ICACRO-C	6.94	8.09	6.23	6.98	5.44	6.74
🟪 ICACRO-I	9.1	14.53	10.89	12.28	9.21	11.2

Figure 9. Improvement percentages of the solutions obtained by the SRS compared to the other four algorithms for different-sized problems P1 to P4, together with the average improvement percentage.

In addition, the mean average improvement of the SRS over Niching PSO was 87.29%, whereas for the SRS over ICA, it was 47.95%, for the SRS over ICACRO-I it was 11.2%, and for the SRS over ICACRO-C it was 6.74%. Hence, it can be understood that the SRS is successful in generating closer-to-optimal solutions compared to the other four algorithms. This underscores the efficiency, scalability, and acceptable performance of the SRS, which outperformed the other four algorithms regardless of the difficulty of the considered various size problems, in solving the search problems.

4.2. Performance Statistical Tests

Table 4 summarizes the results obtained by running the ANOVA test for the five problems P1–P5, where the α level is selected as 0.05. $F(4,\ 7495)$ for P1 to P5 are, respectively, 10,012.93, 4834.08, 8786.23, 6395.4, and 6372.45, which are greater than the F-criteria value of 2.37. In addition, the calculated p-values by this test for all the five problems are 0. This is strong evidence for rejecting the null hypothesis, showing the significant difference between the results of each five problems.

Table 4. The one-way ANOVA test results for different-sized problems P1 to P5.

No.	Problem Size	Source of Variation	Sum of Squares (SS)	df	Mean Square (MS)	F	p-Value	F Criteria
P_1	100	Between Groups	574,774.3	4	143,693.6	10,012.93	0.000000	2.373117
		Within Groups	107,559.2	7495	14.3508			
P_2	200	Between Groups	2,223,419	4	555,854.7	4834.082	0.000000	2.373117
		Within Groups	861,824.6	7495	114.9866			
P_3	300	Between Groups	7,633,617	4	1,908,404	8786.23	0.000000	2.373117
		Within Groups	1,627,944	7495	217.204			
P_4	400	Between Groups	9,107,241	4	2,276,810	6395.4	0.000000	2.373117
		Within Groups	2,668,276	7495	356.0075			
P_5	500	Between Groups	13,354,953	4	3,338,738	6372.445	0.000000	2.373117
		Within Groups	3,926,883	7495	523.9336			

Therefore, as the repeated measures analysis of variance verified the existence of significant differences, we proceeded to perform post hoc procedures in order to pairwise compare the results of the SRS algorithm with those of Niching PSO, ICA, ICACRO-I, and ICACRO-C to identify the significantly different group. The results of the Bonferroni-corrected test is presented in Table 5, in which the Bonferroni-corrected α level is 0.05/5 = 0.01, the * symbol over the p-values stands for p-value < 0.05, and the ** symbol denotes p-value < 0.01. Accordingly, it can be concluded that the results are significantly different, and the null hypothesis is rejected with the confidence level of 99%.

The calculated p-values for the post hoc procedures shown in Table 5 indicate that the differences of the results of the SRS with all the other comparative algorithms can be considered statistically significant with a confidence level of 99%. The most statistically significant differences are found between the results of the SRS and those of Niching PSO and ICA. Then, the smaller but still significant differences lie between the SRS and ICACRO-I and ICACRO-C, where the least differences in these cases are observed in the case of the P4 problem, while the highest one for ICACRO-I is in P3 and for ICACRO-C is in P2. Noting the trends of results of the test, there seems to be no obvious relation between the size of the problem and obtaining statistically significant different results by the SRS. In other words, the size of the search space does not reduce the efficient performance of the SRS.

Table 5. Results of Bonferroni pairwise comparisons.

No.	Problem Size	Groups	N	Mean	Std	*p*-Value with SRS
P_1	100	Niching PSO	1500	58.73	0.89	0 **
		ICA	1500	46.21	1.76	0 **
		ICACRO-C	1500	36.48	3.54	0.000984 **
		ICACRO-I	1500	37.35	3.69	9.41×10^{-15} **
		SRS	1500	35.86	6.45	–
P_2	200	Niching PSO	1500	122.2	1.2	0 **
		ICA	1500	104.87	4.43	0 **
		ICACRO-C	1500	79.76	11.61	0.001799 **
		ICACRO-I	1500	83.35	10.74	3.53×10^{-23} **
		SRS	1500	78.07	17.43	–
P_3	300	Niching PSO	1500	199.31	1.32	0 **
		ICA	1500	150.01	6.8	0 **
		ICACRO-C	1500	117.73	16.07	0.002513 **
		ICACRO-I	1500	120.59	15.38	2.48×10^{-12} **
		SRS	1500	115.51	23.31	–
P_4	400	Niching PSO	1500	258.41	0.87	0 **
		ICA	1500	208.22	8.67	0 **
		ICACRO-C	1500	169.87	21.27	3.55×10^{-7} **
		ICACRO-I	1500	176.23	20.20	3.02×10^{-33} **
		SRS	1500	165.12	29.04	–
P_5	500	Niching PSO	1500	324.71	1.07	0 **
		ICA	1500	259.33	11.39	0 **
		ICACRO-C	1500	217.19	25.08	8.52945×10^{-5} **
		ICACRO-I	1500	221.51	24.09	4.92076×10^{-15} **
		SRS	1500	212.75	35.76	–

* *p*-value < 0.05. ** *p*-value < 0.01.

5. Conclusions and Future Work

In this paper, we considered the novel nature-inspired SRS algorithm, proposed in [9,10], and evaluated its performance in solving a particular CCSC problem, namely the STCOCCSC problem by comparing it with the results of some previously studied well-proposed search algorithms, including ICA, ICACRO-I, ICACRO-C, and Niching PSO.

Experimental results obtained by the SRS over five independent different-sized STCOCCSC problems ranked the SRS as the best-performing algorithm compared to the mentioned four algorithms. A careful improvement percentage analysis of the SRS indicated that from an averaged viewpoint, the SRS solutions were notably better than Niching PSO and ICA solutions with 87.29% and 47.95% improvement percentages, respectively, while the ICACRO-I and ICACRO-C solutions were, respectively, 11.2% and 6.74% improved. In addition, the one-way ANOVA and Bonferroni-corrected post hoc statistical tests confirmed the significance of these improvements.

From a general perspective: (i) The SRS outperformed the four comparative algorithms, ICACRO-C, ICACRO-I, ICA, and Niching PSO, and yielded 6.74, 11.2, 47.95, and 87.29 percent performance improvement on average for five different-sized problems. (ii) The SRS demonstrated a statistically significant superiority with a confidence level of 99% compared to the four comparative algorithms. (iii) No efficiency reduction was observed for the SRS facing higher-dimensional search space problems. In addition, considering the trend of the obtained results at 1500 iterations, the SRS appeared to be more qualified to reach improved solutions with a higher number of iterations, avoiding premature convergence. Accordingly, the SRS can be employed for solving the CCSC problem as well as the other similar-structured optimization problems.

Author Contributions: Methodology, N.K.N.; software, N.K.N.; validation, E.S. and M.A.; formal analysis, N.K.N.; investigation, N.K.N.; resources, N.K.N. and E.S.; writing—original draft preparation, N.K.N.; writing—review and editing, N.K.N., E.S., and M.A.; visualization, N.K.N.; supervision, E.S. and M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Universiti Kebangsaan Malaysia, Research University grant numbered DIP-2018-041, grant numbered FRGS/1/2014/ICT07/UKM/02/1, and grant numbered FRGS/1/2022/ICT11/UKM/02/02.

Data Availability Statement: Not applicable.

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

References

1. Ying, G.-G.; Yu, X.-Y.; Kookana, R.S. Biological degradation of triclocarban and triclosan in a soil under aerobic and anaerobic conditions and comparison with environmental fate modelling. *Environ. Pollut.* **2007**, *150*, 300–305. [CrossRef]
2. Knuth, D. Sorting and searching. *Art Comput. Program.* **1998**, *3*, 513.
3. Singer, B.; Veloso, M. Stochastic search for signal processing algorithm optimization. In Proceedings of the 2001 ACM/IEEE Conference on Supercomputing, Denver, CO, USA, 10–16 November 2001; p. 22.
4. Dorigo, M. Ant colony optimization. *Scholarpedia* **2007**, *2*, 1461. [CrossRef]
5. 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.
6. Jung, J.K.H.M.; McCouch, S.R.M. Getting to the roots of it: Genetic and hormonal control of root architecture. *Front. Plant Sci.* **2013**, *4*, 186. [CrossRef]
7. Atashpaz-Gargari, E.; Lucas, C. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. In Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 4661–4667.
8. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
9. Naseri, N.K.; Sundararajan, E.A.; Ayob, M.; Jula, A. Smart Root Search (SRS): A Novel Nature-Inspired Search Algorithm. *Symmetry* **2020**, *12*, 2025. [CrossRef]
10. Naseri, N.K.; Sundararajan, E.; Ayob, M.; Jula, A. Smart Root Search (SRS): A New Search Algorithm to Investigate Combinatorial Problems. In Proceedings of the 2015 Seventh International Conference on Computational Intelligence, Modelling and Simulation (CIMSim), Kuantan, Malaysia, 27–29 July 2015; pp. 11–16.
11. Jula, A.; Sundararajan, E.; Othman, Z. Cloud computing service composition: A systematic literature review. *Expert Syst. Appl.* **2014**, *41*, 3809–3824. [CrossRef]
12. Cao, B.; Sun, Z.; Zhang, J.; Gu, Y. Resource Allocation in 5G IoV Architecture Based on SDN and Fog-Cloud Computing. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 3832–3840. [CrossRef]
13. Kavre, M.S.; Gardas, B.B.; Narwane, V.S.; Navimipour, N.J.; Yalçın, Ş. Evaluating the Effect of Human Factors on Big Data Analytics and Cloud of Things Adoption in the Manufacturing Micro, Small, and Medium Enterprises. *IT Prof.* **2022**, *24*, 17–26. [CrossRef]
14. Li, T.; Xia, T.; Wang, H.; Tu, Z.; Tarkoma, S.; Han, Z.; Hui, P. Smartphone App Usage Analysis: Datasets, Methods, and Applications. *IEEE Commun. Surv. Tutor.* **2022**, *24*, 937–966. [CrossRef]
15. Jula, A.; Othman, Z.; Sundararajan, E. A hybrid imperialist competitive-gravitational attraction search algorithm to optimize cloud service composition. In Proceedings of the 2013 IEEE Workshop on Memetic Computing (MC), Singapore, 16–19 April 2013; pp. 37–43.
16. Bartz-Beielstein, T.; Chiarandini, M.; Paquete, L.; Preuss, M. *Experimental Methods for the Analysis of Optimization Algorithms*; Springer: Berlin/Heidelberg, Germany, 2010.

17. Dodig-Crnkovic, G. Computer Science in a Theory of Science Discourse. Ph.D. Thesis, Mälardalen University, Västerås, Sweden, 2002.
18. Michalewicz, Z.; Fogel, D.B. Constraint-Handling Techniques. In *How to Solve It: Modern Heuristics*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 231–270.
19. Talbi, E.-G. *Metaheuristics: From Design to Implementation*; John Wiley & Sons: Hoboken, NJ, USA, 2009; Volume 74.
20. Dang, W.; Guo, J.; Liu, M.; Liu, S.; Yang, B.; Yin, L.; Zheng, W. A Semi-Supervised Extreme Learning Machine Algorithm Based on the New Weighted Kernel for Machine Smell. *Appl. Sci.* **2022**, *12*, 9213. [CrossRef]
21. Lu, S.; Guo, J.; Liu, S.; Yang, B.; Liu, M.; Yin, L.; Zheng, W. An Improved Algorithm of Drift Compensation for Olfactory Sensors. *Appl. Sci.* **2022**, *12*, 9529. [CrossRef]
22. Hashemi, M.; Javaheri, D.; Sabbagh, P.; Arandian, B.; Abnoosian, K. A multi-objective method for virtual machines allocation in cloud data centres using an improved grey wolf optimization algorithm. *IET Commun.* **2021**, *15*, 2342–2353. [CrossRef]
23. Ni, Q.; Guo, J.; Wu, W.; Wang, H. Influence-Based Community Partition with Sandwich Method for Social Networks. *IEEE Trans. Comput. Soc. Syst.* **2022**, 1–12. [CrossRef]
24. Tarawneh, H.; Alhadid, I.; Khwaldeh, S.; Afaneh, S. An Intelligent Cloud Service Composition Optimization Using Spider Monkey and Multistage Forward Search Algorithms. *Symmetry* **2022**, *14*, 82. [CrossRef]
25. Singh, M.P. Physics of service composition. *IEEE Internet Comput.* **2001**, *5*, 6–7. [CrossRef]
26. Schmid, S.; Chart, T.; Sifalakis, M.; Scott, A. Flexible, Dynamic, and Scalable Service Composition for Active Routers. In Proceedings of the IFIP-TC6 4th International Working Conference on Active Networks, Zurich, Switzerland, 4–6 December 2002; pp. 253–266.
27. Kosuga, M.; Kirimoto, N.; Yamazaki, T.; Nakanishi, T.; Masuzaki, M.; Hasuike, K. A multimedia service composition scheme for ubiquitous networks. *J. Netw. Comput. Appl.* **2002**, *25*, 279–293. [CrossRef]
28. Milanovic, N.; Malek, M. Current solutions for Web service composition. *IEEE Internet Comput.* **2004**, *8*, 51–59. [CrossRef]
29. Kofler, K.; ul Haq, I.; Schikuta, E. A Parallel Branch and Bound Algorithm for Workflow QoS Optimization. In Proceedings of the Parallel Processing, 2009, ICPP '09 International Conference on Parallel Processing, Vienna, Austria, 22–25 September 2009; pp. 478–485.
30. Zeng, C.; Guo, X.A.; Ou, W.J.; Han, D. Cloud Computing Service Composition and Search Based on Semantic. In *Cloud Computing, Proceedings*; Jaatun, M.G., Zhao, G., Rong, C., Eds.; Lecture Notes in Computer Science; Springer: Berlin, Germeny, 2009; Volume 5931, pp. 290–300.
31. Liu, L.; Zhu, H.; Chen, S.; Huang, Z. Privacy regulation aware service selection for multi-provision cloud service composition. *Future Gener. Comput. Syst.* **2022**, *126*, 263–278. [CrossRef]
32. Jula, A.; Nilsaz, H.; Sundararajan, E.; Othman, Z. A New Dataset and Benchmark for Cloud Computing Service Composition. In Proceedings of the 2014 5th International Conference on Intelligent Systems, Modelling and Simulation, Langkawi, Malaysia, 27–29 January 2014; pp. 83–86.
33. Amiri, Z.; Heidari, A.; Navimipour, N.J.; Unal, M. Resilient and dependability management in distributed environments: A systematic and comprehensive literature review. *Clust. Comput.* **2022**, 1–36. [CrossRef]
34. Han, J.; Kamber, M.; Pei, J. *Data Mining: Concepts and Techniques*; Morgan Kaufmann Publishers Inc.: Burlington, MA, USA, 2011; p. 696.
35. Neapolitan, R.E.; Neapolitan, R.; Naimipour, K. *Foundations of Algorithms*; Jones & Bartlett Learning: Burlington, MA, USA, 2011.
36. López-Bucio, J.; Cruz-Ramırez, A.; Herrera-Estrella, L. The role of nutrient availability in regulating root architecture. *Curr. Opin. Plant Biol.* **2003**, *6*, 280–287. [CrossRef]
37. Linkohr, B.I.; Williamson, L.C.; Fitter, A.H.; Leyser, H.O. Nitrate and phosphate availability and distribution have different effects on root system architecture of Arabidopsis. *Plant J.* **2002**, *29*, 751–760. [CrossRef]
38. Jiang, C.; Gao, X.; Liao, L.; Harberd, N.P.; Fu, X. Phosphate starvation root architecture and anthocyanin accumulation responses are modulated by the gibberellin-DELLA signaling pathway in Arabidopsis. *Plant Physiol.* **2007**, *145*, 1460–1470. [CrossRef] [PubMed]
39. Bates, T.R.; Lynch, J.P. The efficiency of Arabidopsis thaliana (Brassicaceae) root hairs in phosphorus acquisition. *Am. J. Bot.* **2000**, *87*, 964–970. [CrossRef]
40. Rubio, V.; Bustos, R.; Irigoyen, M.L.; Cardona-López, X.; Rojas-Triana, M.; Paz-Ares, J. Plant hormones and nutrient signaling. *Plant Mol. Biol.* **2009**, *69*, 361–373. [CrossRef]
41. López-Bucio, J.; Hernández-Abreu, E.; Sánchez-Calderón, L.; Nieto-Jacobo, M.F.; Simpson, J.; Herrera-Estrella, L. Phosphate availability alters architecture and causes changes in hormone sensitivity in the Arabidopsis root system. *Plant Physiol.* **2002**, *129*, 244–256. [CrossRef]
42. Nacry, P.; Canivenc, G.; Muller, B.; Azmi, A.; Van Onckelen, H.; Rossignol, M.; Doumas, P. A role for auxin redistribution in the responses of the root system architecture to phosphate starvation in Arabidopsis. *Plant Physiol.* **2005**, *138*, 2061–2074. [CrossRef]
43. Bates, T.; Lynch, J.P. Stimulation of root hair elongation in Arabidopsis thaliana by low phosphorus availability. *Plant Cell Environ.* **1996**, *19*, 529–538. [CrossRef]
44. Lipowski, A.; Lipowska, D. Roulette-wheel selection via stochastic acceptance. *Phys. A Stat. Mech. Its Appl.* **2012**, *391*, 2193–2196. [CrossRef]
45. Blum, A. *Plant Breeding for Water-Limited Environments*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2010.

46. Hultine, K.; Cable, W.; Burgess, S.; Williams, D. Hydraulic redistribution by deep roots of a Chihuahuan Desert phreatophyte. *Tree Physiol.* **2003**, *23*, 353–360. [CrossRef]
47. Prieto, I.; Armas, C.; Pugnaire, F.I. Hydraulic lift promotes selective root foraging in nutrient-rich soil patches. *Funct. Plant Biol.* **2012**, *39*, 804–812. [CrossRef]
48. Bates, T.R.; Lynch, J.P. Plant growth and phosphorus accumulation of wild type and two root hair mutants of Arabidopsis thaliana (Brassicaceae). *Am. J. Bot.* **2000**, *87*, 958–963. [CrossRef]
49. Jula, A.; Sundararajan, E.A.; Othman, Z.; Naseri, N.K. Color Revolution: A Novel Operator for Imperialist Competitive Algorithm in Solving Cloud Computing Service Composition Problem. *Symmetry* **2021**, *13*, 177. [CrossRef]
50. Zibin, Z.; Yilei, Z.; Lyu, M.R. Distributed QoS Evaluation for Real-World Web Services. In Proceedings of the Web Services (ICWS), 2010 IEEE International Conference on Web Services, Miami, FL, USA, 5–10 July 2010; pp. 83–90.
51. Little, R.J.; Rubin, D.B. *Statistical Analysis with Missing Data*, 2nd ed.; John Wiley & Sons: New York, NY, USA, 2002.
52. Sterne, J.A.C.; White, I.R.; Carlin, J.B.; Spratt, M.; Royston, P.; Kenward, M.G.; Wood, A.M.; Carpenter, J.R. Multiple imputation for missing data in epidemiological and clinical research: Potential and pitfalls. *Res. Methods Report.* **2009**, *339*, 157–160. [CrossRef] [PubMed]
53. Royston, P.; Whit, I.R. Multiple Imputation by Chained Equations (MICE): Implementation in Stata. *J. Stat. Softw.* **2011**, *45*, 1–20. [CrossRef]
54. Liao, J.X.; Liu, Y.; Wang, J.Y.; Zhu, X.M. Service Composition Based on Niching Particle Swarm Optimization in Service Overlay Networks. *Ksii Trans. Internet Inf. Syst.* **2012**, *6*, 1106–1127. [CrossRef]
55. Liao, J.X.; Liu, Y.; Zhu, X.M.; Xu, T.; Wang, J.Y. Niching Particle Swarm Optimization Algorithm for Service Composition. In Proceedings of the 2011 IEEE Global Telecommunications Conference, Houston, TX, USA, 5–9 December 2011; IEEE: New York, NY, USA, 2011.
56. Nakagawa, S. A farewell to Bonferroni: The problems of low statistical power and publication bias. *Behav. Ecol.* **2004**, *15*, 1044–1045. [CrossRef]
57. Cabin, R.; Mitchell, R. To Bonferroni or not to Bonferroni: When and how are the questions. *Bull. Ecol. Soc. Am.* **2000**, *81*, 246–248. [CrossRef]
58. Holm, S. A Simple Sequentially Rejective Multiple Test Procedure. *Scand. J. Stat.* **1979**, *6*, 65–70. [CrossRef]