Neural Computing and Applications (2021) 33:409–432
https://doi.org/10.1007/s00521-020-04983-8

ORIGINAL ARTICLE

Spiking neural firefly optimization scheme for the capacitated dynamic vehicle routing problem with time windows

Resmi RamachandranPillai[1] · Michael Arock[1]

Received: 18 September 2019 / Accepted: 2 May 2020 / Published online: 20 May 2020
© Springer-Verlag London Ltd., part of Springer Nature 2020

Abstract
A number of technological improvements have prompted a great concern on 'dynamism' in vehicle routing problems (VRP). In real-world applications, the dynamic information happens simultaneously with the plan being carried out. In order to effectively solve dynamic VRP (DVRP), many optimization strategies have been introduced in the literature. A new variant of vehicle routing problem is proposed which combines DVRP with time windows and capacity constraints, named capacitated DVRP with time windows (CDVRPTW). Apart from the traditional way of handling the problem, this paper proposes a novel strategy that incorporates improved firefly algorithm (IFA) into the framework of spiking neural *P* (SN *P*) systems, named spiking neural firefly optimization (SFO). A mathematical model of the problem is formulated, and the solution scheme is designed by associating a number of SN *P* systems that work in parallel to find optimal solutions in a reasonable time. Additionally, the parameters in the IFA are optimized by adjusting the rule probabilities using SN *P* systems. Being a NP-hard problem with real-world applications, the benefits of this study are far-reaching. The proposed scheme has been tested on benchmark instances and proved novelty, feasibility, and potentiality of the system.

Keywords Spiking neural *P* systems · Firefly optimization · Dynamic vehicle routing · Membrane computing

1 Introduction

Natural computing (NC) [1] refers to the process inspired by structure, function, and behavior of biological as well as natural systems. There are many variants of NC, which include artificial neural networks (ANN) [2], swarm intelligence [3], evolutionary algorithms [4], quantum computing [5], and membrane computing [6]. Membrane computing (MC) is a branch of computer science that aims to find new computational and mathematical models from the structure and functioning of cellular membranes. The models considered under MC are distributed and parallel systems called *P* systems handling multisets of objects in the cells defined by placing membranes hierarchically or generally.

Many engineering design optimization problems are solved using MC in a reasonable time because of its nondeterminism and maximum parallelism features [7]. The arrangement of membranes categorizes the MC systems into cell-like, tissue-like, and neural-like systems. The latter class is the most recent branch of MC, which is incorporated by spiking neurons into the area of *P* systems (SN *P* systems). SN *P* systems fall under the third generation of artificial neural network models. These systems work by the spiking nature of neurons.

SN *P* systems can be represented by a directed graph with nodes referring to spiking neurons, and edges form synapses between neurons through which communication takes place. SN *P* systems with neuron division and dissolution [8], budding [9], and spikes and antispikes [10] are applied in numerous applications and proven to be efficient and reliable. Earlier SN *P* systems were treated as language-generative devices of 0's and 1's [11]. An optimization SN *P* system to find the solutions of an unconstrained single objective optimization problem has been proposed to solve knapsack problems [12]. Spike train

✉ Resmi RamachandranPillai
resmiramachandranpillai@gmail.com; 405116006@nitt.edu

Michael Arock
michael@nitt.edu

[1] Department of Computer Applications, National Institute of Technology, Tiruchirappalli, India

and timing are the two key factors of SN *P* systems that are used for spiking and coding of information [13].

Because of the better theoretical foundations of MC, many intelligent systems employ new methods and paradigms [14]. It is used to identify the nuclear export signals of amino acid sequence, which is a challenge in computational biology [15]. Recently, SN *P* systems are employed in many intelligent and expert domains such as the semantics of deductive database systems [16] and parallel multiples [17]. So it is concluded that SN *P* systems are more powerful in intelligent and expert systems from both theoretical and practical sides [18].

An optimization problem [19] is a problem of finding the best solution from a set of feasible solutions that cannot be solved in polynomial time limit using any deterministic method. Numerous methods and schemes are designed to solve such kinds of problems [20–22]. Most of the existing schemes in the literature emphasize on arriving at the optimal solutions quickly rather than focusing on the time limit. One of the objectives of this paper focuses on the balance between arriving the best solution and reducing the time limit. Apart from that, a novel way of choosing, controlling, and activating spikes is employed.

Capacitated VRP with time windows (CVRPTW) [23, 24] is a variant of VRP, which has emerged due to the increasing demands of customers in the specific time span. It is considered as a nondeterministic polynomial hard (NP-hard) problem that finds an ideal arrangement of routes to be serviced by a fleet of vehicles of some fixed capacity to serve a given arrangement of customers inside the allotted time windows. The customer's geographic locations and demands are unpredictable while designing routes for a particular vehicle. In this way, it is critical to make a decision on a choice on the allocation and scheduling of new demands by considering all the components that are significantly influenced.

The intention of this paper is to design a spiking neural framework to solve CVRPTW with dynamism (CDVRPTW) in a feasible time without sacrificing the global optimum solutions. The proposed solution scheme is designed by combining a number of SN *P* systems coupled with an improved firefly optimization algorithm (SFO). It is based on the flashing behavior of fireflies, which is a signal system to attract other fireflies. As noticed in the literature [25–27], the firefly algorithm (FA) has been applied to many engineering design optimization problems. Only limited articles exist which combine FA and VRP [28–31]. The strong adaptability and robustness, easy to set up, minimal manual adjustments, and minimal parameters made it suitable for VRP problems. But, the reason behind its minimal usage in VRP kind of problems is that it is prone to local optima. But, in the proposed method, the authors overcome it by carefully designing the initial assignments of fireflies and parameter optimization.

The main contributions of the proposed scheme can be stated as:

1. We propose a spiking neural-based firefly optimization scheme (SFO) to find solutions to CDVRPTW.
2. The firefly optimization is modified to explore the solution space faster, and thereby achieving a high convergence rate.
3. Parameters in the proposed firefly scheme are optimized by adjusting the rule probabilities in the SN P systems.
4. A notion of movement and attraction of fireflies is introduced by taking real-world circumstances like traffic data and dynamic requests.
5. A high degree of parallelism is incorporated in the cluster level by considering solutions together and assigning them to different spiking neurons for simultaneous computation. Here, cluster means a group of customers who are geographically mapped by distance.
6. The initial feasible positions for fireflies are carefully designed using Clarke and Wright [32] algorithm rather than the random assignment.
7. A mathematical formulation of the problem is made with real-world constraints.

The remaining sections are organized as follows: Sect. 2 carries out a detailed literature review on the application and solution approaches to the problem. Section 3 details the problem description and mathematical formulation of CDVRPTW. Section 4 outlines a basic SN *P* system. Section 5 explains the improved firefly optimization scheme using SN *P* systems. Section 6 describes the proposed system design and the various modules involved. Section 7 talks about the experimental results and discussions in detail. Concluding comments and future scope of the system are given in Sect. 8.

2 Literature review

The advancement of technology in various fields has promoted the evolution of intelligent transportation systems (ITS), which uses the history of geolocation techniques with geographic information systems [33]. Vehicle routing is aiming for an operational task that is bound to the competence of dispatches as well as the optimization costs which are directly reliant on the size of the fleet [34–38]. The applications of routing in various aspects of life have been found.

2.1 Transport of goods, services, and peoples

Because of the highly variable traveling time, one should take factors like traffic, competition, and cooperation between transport companies into account [39, 40]. All these applications come under city logistics. Planning real-time traffic and dynamic routing of a fleet of vehicles, which require additional modules and attributes, are actualized [41]. A decision support system (DSS) can be used for a better trade-off between dynamic travel and service times.

In courier transportation systems, one should take into account not only the request locations, time window, and capacity constraints but also traffic data and forbidden paths. Optimization-enabled automatic fleet management system (AFMS) has considered for solving such types of problems [42–44].

Customer satisfaction is an important factor when we define an FMS. Newspaper delivery is an example of such type of domains where the complaints will be filed in case of delay of delivery. In order to reduce costs and improving customer satisfaction, centralized applications were proposed that made use of cell phones to continuously communicate with drivers at the same time of performing routing [45]. Additionally, future requests are anticipated using historical data of customers [46].

In grocery delivery services, the merchant designs various clients that can be considered inside a fixed time window. At the same time, it is made inaccessible to the clients when the capacity constraints are violated. The time windows fixed for a customer are dynamically planned based on future request in-home delivery problems [47]. Greedy randomized adaptive search procedure (GRASP) and adaptive large neighborhood search (ALNS) were proposed which consider uncertainty by introducing scenarios for possible demand realizations [48]. Dynamic stacker crane problems [49, 50] which considered container carriers with loading and unloading ships are coming under operations research applications. In addition, factories and hospitals are other application domains where products or medical instruments, respectively, must be transferred [51].

Transport of peoples is characterized by additional constraints such as regulation on waiting, service, and travel times. Cab system is the most common online individual transportation system where the customer's requests are composed of pick time and location coupled with the dropping location. Many variants are available like advance booking, sharing, etc., which leaves a limited chance for optimization. The multicab metropolitan transportation system is focused on where more than one customer's request can be serviced [52]. Other application domains include transport of children to schools, disabled people to work locations, patients to medical centers, etc. [53–55]. Air taxis are other domains where the limitations of the traditional airline are reduced.

2.2 Solution approaches

Ranging from linear programming (LP) to meta-heuristics, there are many schemes, which solve DVRP and its variants.

2.2.1 Dynamic and deterministic synchronous optimization techniques

The timely change of critical information forces new hindrances in the absence of stochastic information. In this specific situation, exact methods are irrelevant since the optimal solutions are time-dependent. So most of the approaches on DVRP rely on heuristics that find the best solutions to the current instance. The first optimization focused on the dynamic arc routing problem (DARP), which reconfigures the route each time a new request has been made. The main drawback was the lack of dimensionality, which prevents its application to large instances. Decision epochs or time slices are later introduced in the literature. All these schemes relied on the algorithms of static routing, but the main drawback was the updating of routing, which increases delays at the customers' ends [56, 57].

A real-time truckload pickup and delivery (PDP) was addressed [58], where a fleet of trucks has to serve requests arriving dynamically. It is proposed a rolling horizon approach based on LP. A dynamic column generator scheme for DVRPTW has been addressed [59]. The experimental results based on Solomon's benchmark instances reveal that it yields better results in terms of fitness function, but time for performing computation was very high.

An ant colony system (ACS) [60] to solve DVRP was developed by using the time slice strategy. At each time slice, the requests are checked and are handled till the end of that time slice. They used pheromone trace to transfer good solutions to the next time window. Similar approaches were also addressed in [61, 62]. The main drawback of this scheme is that the intensity of pheromone concentration may vanish as the number of iterations increases. It may lead to having uncertain time to converge for larger instances.

2.2.2 Dynamic and deterministic continuous optimization techniques

These kinds of strategies perform optimization throughout and keep information about good solutions in memory [63].

When a dynamic request has been made, it uses the memory where the previous information was stored to plan and serve new requests. Therefore, the computational capacity is maximized. In addition, the vehicles are unaware of the next requests to be served until they complete the service of the previous request. The introduction of parallel tabu search (TS) was the first continuous optimization approach to courier services. A pool of good route and customer pairs is maintained in memory, and this information was referred for designing initial feasible solutions. At the point when a client makes a request, it is checked in the memory to choose whether it ought to be accepted or discarded. Different variants like DVRP and DARP have additionally been implemented using a similar scheme [64, 65]. The main drawback of the TS is that, even though it is semi-deterministic, it produces fewer quality solutions in the long run because of the hard constraint settings.

A generalization of TS using a multiple plan approach (MPA) was designed to populate and maintain routing plans to generate a distinguished solution. Pool updates were performed whenever a vehicle finishes the serving of a customer. Nevertheless, it was very crucial in high-dimensional problems. Genetic algorithms (GA) with a human dispatcher to study the D-PDP variant were proposed [66, 67]. The measure of dynamism was low for GA in spite of the fact that it keeps running all through the horizon and continually updating the changes.

2.2.3 Dynamic and stochastic schemes

A truckload PDP based on the Markov decision process (MDP) was formulated [68]. A notion of probability has then been added to solve VRP along with MDP [69]. However, it lacked dimensionality and was not suitable for real-world applications. The approximate dynamic program (ADP) was designed in order to reduce the limitations of traditional DP [70]. It was successfully applied to FMS [35, 37]. But, the main limitation of ADP was that a single observation may change the whole value function, thereby modifying the constraints which further distorts successive observations. The LP with dynamic and stochastic content has also been designed [58, 71]. Later, emergency vehicle dispatching and routing was developed and was used by Haghani and Yang [72]. The problems with these schemes were again the curse of dimensionality.

2.2.4 Sampling

It relies on the generation of scenarios. A scenario is an outline of known and future events at each time unit on the horizon. Multiple scenario approach (MSA) has been connected to Solomon's benchmark instances [73] and performed substantially better than the state-of-the-art algorithms. The MSA with a consensus scheme has been used to tackle DVRP [74], and an event-driven framework with MSA [75] has been implemented later. Dynamic sample scenario hedge heuristics (DSHH) divides the horizon into intervals. Routing is done by assigning a subset of requests to the vehicles relying upon the frequencies of their assignments [76]. It later promoted the design of branch and regret heuristics. Though all of the above schemes performed well with dynamism, maintaining and modifying scenarios at each time units added complexity of the system.

2.2.5 Other strategies

Waiting strategy was introduced in [77] in order to serve dynamic requests in the neighborhood of a served request and plan accordingly [78–82]. Relocation strategy was also designed in case of emergency vehicle routing problems and is applied in D-VRP, DVRPTW, DTSPTW, etc. [83]. All these strategies were based on delaying the customer's request assignments to vehicles in a priority manner. So some customers may enter a long waiting period which in turn affects the overall system performance. Also, an adaptive spiking neural P system [84] is designed to handle VRP problems, but it did not discuss any dynamic requests and assignments of customers.

Due to the high complexity of VRP and its variants, state-of-the-art algorithms are very far from the real-time requirements in terms of quality and efficiency. Therefore, there are many factors that need to be considered such as how to find feasible solutions, how to avoid local optimum, how to control the convergence rate, and how to optimize the problem within the acceptable range. The proposed scheme presents an improved firefly optimization scheme using SN *P* systems to find solutions for CDVRPTW. It incorporates optimization, parallelism, and determinism into the SN *P* system framework. To the best of the author's knowledge, this is the first work of its kind that solves CDVRPTW by taking advantage of both FA and SN *P* systems with the intelligent exploration and exploitation of solution space. Analysis of the experimental outcomes justifies the novelty of the proposed scheme.

3 Problem description and mathematical model

In this section, a brief description of CDVRPTW will be explained together with the mathematical formulation of the problem. The measures for dynamism are also explained.

3.1 Capacitated dynamic vehicle routing problems with time windows (CDVRPTW)

The CDVRPTW is a variant of VRP that has emerged because of recent improvements in real-world communications. Larsen [85] introduced two aspects of DVRP: Not all information on the routing process is available when it begins; information may change even after the routes have been constructed.

To understand the problem, Fig. 1 illustrates the route planning of a CDVRPTW. Here, there are 3 vertices (A, B, and C) that correspond to 3 different customers with known requests at $t = t_0$. The edges connecting the customers denote different parameters like traveling cost, traveling distance, etc., that are to be optimized. The initial route assignment to serve known requests (Depot, A, B, C, Depot) is planned before the vehicle leaves the system. While the vehicle follows the route AB, one dynamic request appeared at $t = t_1$ (Fig. 2) in that cluster. So, the vehicle needs to reroute its path as (Depot, A, B, C, D, Depot) if the total capacity of the vehicle is greater than or equal to the sum of all capacities of each customer in the route.

3.2 The degree of dynamism

The performance of a DVRP is estimated based on the number of dynamic demands and the time when these demands arise, though static VRP depends on the number of clients and their distribution. The measure of 'dynamism' would be significant when we investigate the performance of a particular algorithm under constraints. The role of the new request during the calculation phase of a routing framework can be used to analyze dynamism. The measure here is the number of dynamic demands with respect to the total demands. This proportion is known as the degree of dynamism and is expressed by [86]:

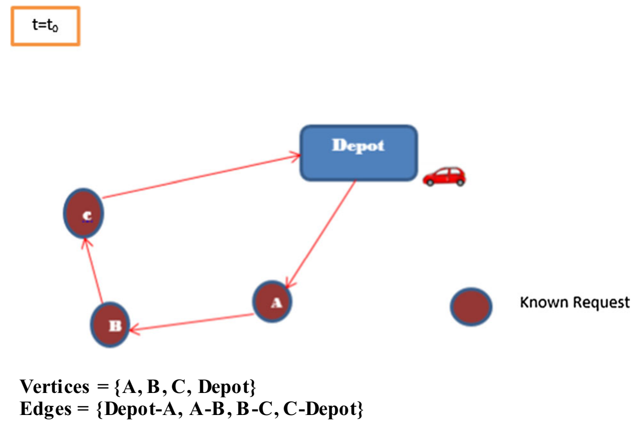

t=t₀
Depot
C
A
B
Known Request
Vertices = {A, B, C, Depot}
Edges = {Depot-A, A-B, B-C, C-Depot}

Fig. 1 Route planning in the static environment (at $t = t_0$)

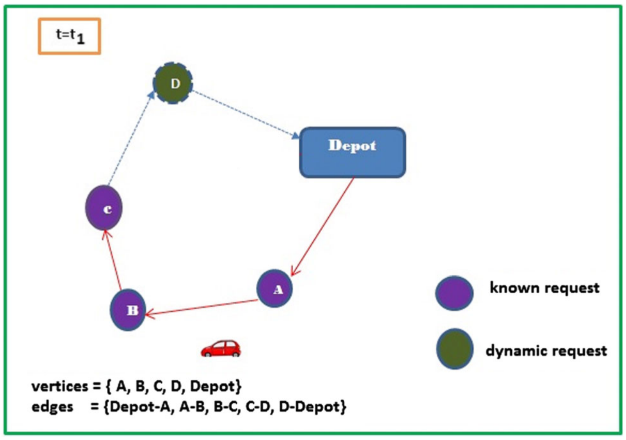

t=t₁
D
Depot
C
A
B
known request
dynamic request
vertices = { A, B, C, D, Depot}
edges = {Depot-A, A-B, B-C, C-D, D-Depot}

Fig. 2 Route planning in the dynamic environment (at $t = t_1$)

$$\text{dod} = \frac{\text{Number of dynamic requests}}{\text{Total number of requests}} \tag{1}$$

This measure relies only on the number of requests but not on the time limits.

Dynamism within fixed time windows has several routing applications where the service must start and finish within a fixed known interval of time. In the case of time windows, dynamism is expressed by effective dod (edod). It is a measure of the average of how later the customer requests are accepted compared to the latest time the requests could be received. Let t_k be the time at which kth request is received, let e_k and l_k, respectively, be the start and end of a particular time window, and the planning horizon is defined between 0 and T such that $0 \leq t_k \leq T$. We define reaction time (r_k) as the distance between the time the request is made and the latest time at which the service should starts. The reaction time of kth request (Fig. 3) is defined by:

$$r_k = l_k - t_k \tag{2}$$

In case of time windows (t_w), the effective dod (e_{dod}) can be defined as:

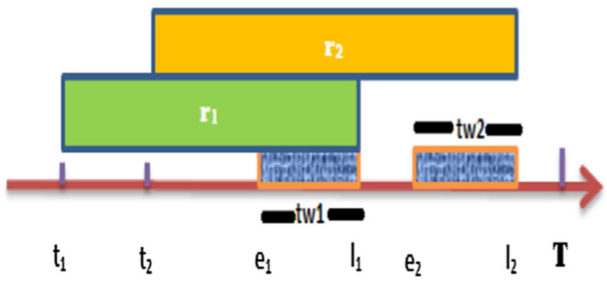


Fig. 3 The reaction time of two customers under time windows

$$e_{\text{dod}} - t_{\text{w}} = \frac{1}{n}\sum_{k=1}^{n}\left(\frac{T-(l_k-t_k)}{T}\right) = \frac{1}{n}\sum_{k=1}^{n}\left(1-\frac{r_k}{T}\right) \tag{3}$$

where $0 \le e_{\text{dod}} - t_{\text{w}} \le 1$ and $l_k - t_k \le T, \quad k = 1,2,\ldots,n$, where n is the sum of immediate requests and advanced requests during the horizon. In the proposed work, the DVRP model is treated as a set of static VRP by the event scheduler. The event scheduler is responsible for receiving customer requests and creating static problems. Then, it sends the problems to the optimization procedure (here SFO) and returns the optimal solutions.

3.3 Mathematical model of the problem

CDVRPTW can be considered as a complete graph-theoretical problem with $G = (V, E)$, where $V = \{V_0 \ldots V_B\}$, the vertex set and $E = \{(V_i, V_j): V_i, V_j \in V \text{ and } i \ne j\}$, the edge set. The vertex V_0 corresponds to the depot. Each vertex $V_k \in V$ has parameters, namely a request P_k, arrival time T_{a_k}, the waiting time T_{w_k}, service time T_{s_k}, and the time window $[T_{e_k}, T_{l_k}]$. C^t_{kj} is the transportation cost from customer j to customer k in the time period t. Each vehicle k has a nonnegative capacity W_k. The total number of time periods is denoted by T, and f_c and t_c, respectively, are the fixed cost and traveling cost/unit time of the vehicle.

$$x_{ki} = \begin{cases} 1, & \text{if customer } k \text{ is delivered by vehicle } i \\ 0, & \text{otherwise} \end{cases}$$
$$y^t_{kj} = \begin{cases} 1, & \text{if vehicle serves the customer } j \text{ from customer } k \text{ in } t \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

The objective of the problem is to minimize the total cost, which is a summation of the fixed cost of vehicles and the routing cost:

$$\text{minimize } F = f_c \times \sum_{k\in V}\sum_{j\in V-\{V_0\}}\sum_{t=1}^{T} y^t_{kj} + t_c \times \sum_{k\in V}\sum_{j\in V-\{V_0\}}\sum_{t=1}^{T}(C^t_{kj} \times y^t_{kj}) \tag{5}$$

subject to the following constraints:

(a) Allocation of customers to vehicles: Each customer has been allotted to only one vehicle at a given time:

$$\sum_{k=0,k\ne j}^{B}\sum_{t=1}^{T} y^t_{kj} = 1, \quad j = 1,2,\ldots,B;$$

$$\sum_{j=1,j\ne k}^{B}\sum_{t=1}^{T} y^t_{kj} = 1, k = 1,2,\ldots,B \tag{6}$$

(b) The capacity of the depot: The number of vehicles departed from the depot does not exceed the total number of vehicles (N) at the depot:

$$\sum_{j=1}^{B}\sum_{t=1}^{T} y^t_{0j} \le N \tag{7}$$

(c) The same vehicle services customers on the same cluster:

$$\sum_{i=1}^{N} N(x_{ki} - x_{ji}) \ge M\left(\sum_{t=1}^{T} y^t_{kj} - 1\right), \quad \forall k,\ j, k \ne j$$
$$\sum_{i=1}^{N} N(x_{ki} - x_{ji}) \le M\left(1 - \sum_{t=1}^{T} y^t_{kj}\right), \quad \forall k,\ j, k \ne j \tag{8}$$

where M is a very large number which is defined as the maximum of the differences between the latest time of kth customer and earliest time of jth (consecutive) customer, $\forall k, j, k \ne j$ in the route.

(d) Every customer node must be serviced by a single vehicle:

$$\sum_{i=1}^{B} x_{jk} = 1, \quad \forall j = 1,2,\ldots,B \tag{9}$$

(e) Vehicle capacity (W_k): The load to deliver to customers by a vehicle should not exceed the vehicle capacity for a customer demand P_k.

$$\sum_{k=1}^{B} P_k x_{ki} \le W_k, \quad \forall i = 1,2,\ldots,N \tag{10}$$

(f) Eliminating alternate path: There should not be any circuit if the vehicle serves the customer j from customer k at t. Here $|A|$ is the total number of customers in the path.

$$\sum_{k,j\in A\times A, k\ne j}\sum_{t=1}^{T} y^t_{kj} \le |A| - 1, A \in \{1,2,\ldots,B\} \quad \forall k,\ j, k \ne j \tag{11}$$

(g) Time window constraints: The constraints on time to service a customer should not exceed the time window.

$$T_{a_k} + T_{w_k} + T_{s_k} + C^t_{kj} - M\left(1 - y^t_{kj}\right) \le T_{aj},$$
$$T_{aj} \le T_{lj},$$
$$T_{w_k} = \max\{T_{e_k} - T_{a_k}, 0\} \tag{12}$$

(h) The values of x_{ki} and y^t_{kj} should be:

$$y_{kj}^t = 0, 1, \quad \forall k, j, i \\ x_{ki} = 0, 1, \quad \forall i, k \tag{13}$$

4 Spiking neural P systems

An SN *P* system [87] of degree $m \geq 1$ can be expressed by a tuple,

$$\Pi = (S, \sigma_1, \ldots, \sigma_m, S_n, O_t)$$

where

(1) $S = \{a\}$ is a singleton alphabet called spike.
(2) $\sigma_1, \ldots, \sigma_m$ are neurons of the form:
$\sigma_i = (Q_j', s_{j,0}, w_{j,0}, R_j)$, $1 \leq j \leq m$, where
- (a) Q_j' is a finite set of states.
- (b) $s_{j,0} \in Q_j$ is the initial state.
- (c) $w_{j,0} \in S^*$ is the initial multiset.
- (d) R_j is a finite set of rules of the form: $s\ w \to s'\ x\ y_{go}\ z_{out}$, where s, $s_0 \in Q_j'$, w, $x \in S^*$, $y_{go} \in (S \times \{go\})^*$, and $z_{out} \in (S \times \{out\})^*$, with the restriction that $z_{out} = \lambda$ for all $j \in \{1, 2, \ldots, m\}$ different from i_0.

(3) $S_n \subseteq \{1, 2, \ldots, m\}\ x\ \{1, 2, \ldots, m\}$ with $(j, j) \notin S_n\ \forall j \in \{1, 2, \ldots, m\}$ are called synapses between neurons.
(4) O_t indicates the set of output neurons σ_o, $o \in \{1, 2, \ldots, m\}$.

The rule set R_i defines the standard rules of the SN *P* system. The term 'go' means the spikes have to leave immediately the neuron to the neighboring neurons through synapses. Those objects that are marked with 'out' leave the system. Thus, when the configuration reaches a state where no rule can be applied, the computation halts.

5 An improved FA based on SN *P* systems

There are a few complications in using the classical FA to solve dynamic VRP problems: (1) The solution space of classical FA is continuous in the real domain, while the solution space of VRP is discrete integer domain. (2) The component γ, the light absorption coefficient plays a key role in the light absorbance. In the event that the estimation of γ is low, at that point it makes FA falls into a local minimum, while a larger estimation of γ forces lower convergence rate. The domain of γ is [0.01, 100] in general.

The step length factor α controls the population diversity, which improves the searching capability and avoids premature convergence. The domain of α is [0, 1]. We adopted the discrete iterative position function to make the solution space suitable for VRP. In addition, the initial feasible positions of fireflies are designed by the Clarke and Wright algorithm which improves the solution space of the problem. Furthermore, the parameters α and β_0 are optimized using SN *P* systems. Also, the light absorption coefficient γ is reinitialized each time the scheme enters stagnation or when no improvements have made for a certain number of consecutive iterations.

The firefly movement can be expressed by:

$$X_i^{\text{new}} = X_i^{\text{old}} + \beta \otimes (X_j \ominus X_i) \oplus \alpha(rand - 0.5) \tag{14}$$

where $X_j \ominus X_i$ denotes the movement direction of the firefly. It is defined as follows:

$$X_j \ominus X_i = \begin{cases} X_j - X_i \notin X_i \\ 0, \quad \text{otherwise} \end{cases} \tag{15}$$

The $\alpha(rand - 0.5)$ and β are used to control the distance between fireflies j and i. Let D_i be the distance of firefly i moving to firefly j and can be defined as:

$$D_i = \begin{cases} X_j \ominus X_i \oplus \alpha(rand - 0.5) < \beta \\ 0, \quad otherwise \end{cases} \tag{16}$$

Definition 1 The brightness of ith firefly can be represented as:

$$I_i = f(i) \tag{17}$$

where $f(i)$ denotes the fitness value of ith firefly which is evaluated by the objective function defined in Eq. 5.

Definition 2 The attraction factors between firefly i and firefly j are $\beta_{(r)} = \beta_0 e^{-\gamma r^2}$, where β_0 is the attraction degree at $\gamma = 0$.

Definition 3 The probability of firefly i to move toward firefly j at the kth dimension is:

$$\text{Let } P_{ij}^k(t) = \frac{\beta_{fk}^{\delta} \left({}^1\!/_{r_{ij}} \right)^{\delta'}}{\sum_{j \notin \text{allowed}} \beta_{fk}^{\delta} \left({}^1\!/_{r_{ij}} \right)^{\delta'}}, \tag{18}$$

where $\beta_{f_k} = \beta_0 e^{-\gamma r_{ij}^2}$

The parameters δ and δ' are used to weight the corresponding terms based on requirements.

If the system attains stagnation, the value of γ is reinitialized. So the value of γ is not fixed, and it varies after a certain number of iterations where there is no progress in the solution space. A firefly will move through the path with a low light absorption coefficient. The smaller the light absorption coefficient, the higher the brightness.

The FA can also be modified to direct fireflies not to blindly follow the distance metric, but also consider the obstacles, traffic data, and dynamic requests coming in between them.

The idea is as follows: Suppose firefly 1 is at position P_1 and firefly 2 is at position P_2. Let AB is any obstacle. In addition, let the brightness of firefly 1 at P_1 is less compared to the brightness of firefly 2 at P_2. Let X and Y are light sources with absorption coefficient γ^1 and γ^2, respectively, and $\gamma^1 < \gamma^2$. Firefly 1 will first choose the random path through Y, as the distance is less compared to the path through X. However, while moving to P_2, the brightness of firefly 1 will be reduced due to the light source at Y and will lead to nonoptimal solutions in future. Even though the distance is high through X, the firefly 1 must select this path since the absorption coefficient is less and will eventually move to better solution space. If we map this scenario to the VRP domain, the light source can be treated as traffic data or dynamic requests that should be taken care of while deciding the route for the vehicles. Figure 4 shows this scenario.

Determination of the parameters α and β_0. The parameters α and β_0 play important roles in FA. By playing out an attentive hunt of the parameter space, the preferred parameter settings those acquired through observational means could be anticipated. The domain of both the parameters is [0, 1]. In order to optimize the parameters using SN P systems, we are considering binary values. Here seven bits are taken to denote every parameter value. If the binary string is 1001111 for α or 79 in decimal representation, the actual value of α is $\frac{79}{127} = 0.62$ in two-point precision, i.e., if the decimal number estimation of the binary string is y, the actual estimation is $\frac{y}{127}$. The same calculations can be applied for β_0 also. So, 127 values are mapped to an interval of width 1. If the number of bits for representing value is high, exactness can be further improved, but the length of the spike train will be increased. The (α, β_0) denotes parameter string, and 3 such strings are connected on each spike train.

$$\underset{\alpha}{1010111}|\underset{\beta_0}{0011111}|\underset{\alpha}{1111111}|\underset{\beta_0}{0000000}|\underset{\alpha}{1000001}|\underset{\beta_0}{0101010}$$

The encodings are:

(α, β_0) = {(87/127, 31/127), (127/127, 0/127), (65/127, 42/127)}

(α, β_0) = {(0.68, 0.24), (1.00, 0.00), (0.51, 0.33)}

An SN P system is designed to produce a binary string, which is used to encode the values of the parameters. Then, it activates the evolution rules based on probability and the output from multiple neurons are collected. A number H of such SN P systems are connected through an adapter, which is responsible for dealing with population and probability adjustment.

The flow diagram of parameter optimization is given in Fig. 5. The family of SN P systems connected through the adapter module is shown in Fig. 6. A graphical representation of IFA using SN P system optimization is depicted in Fig. 7.

Definition 4 An optimization SN P system of degree $m_o \geq 1$ can be formally defined by [88]:

$\Pi = (O, \sigma_1 \ldots \sigma_{m_o}, \sigma_{I1}, \sigma_{I2}, S, I_o)$

1. $O = \{a\}$ is the singleton alphabet.
2. $\sigma_1 \ldots \sigma_{m_o}$ are neurons of the form $\sigma_1 = (1, R_k, P_k)$, and $R_k = \{r_k^1, r_k^2\}$ is the rule set of the form, $r_k^1 = \{a \rightarrow a\}$ $r_k^2 = \{a \rightarrow \lambda\}$ and $P_k = \{P_k^1, P_k^2\}$ is a set of probabilities of rule set R_k, where $P_k^1 + P_k^2 = 1$, $\sigma_{I1} = \sigma_{I2} = (\{1, a \rightarrow a\})$.
3. $S = \{(k, l) \; (1 \leq k \leq m_o + 1 \;\; \Lambda \;\; l = m_o + 2) \bigvee (k = m_o + 2 \bigwedge l = m_o + 1)\}$
4. $I_o = \{1, 2 \ldots m_o\}$ is the set of output neurons.

Generating Initial Solutions First, we need to design initial feasible solutions considering customer allocation and path planning. A random assignment of visit day combination for each customer is planned using the Clarke and Wright savings algorithm. Here, the routes cannot be merged without violating the time duration or capacity constraints. Because of this, the number of vehicles for serving the routes may not be adequate. In such cases, route merging is carried out in which the route with the fewest customer is merged to other nearby routes minimizing the cost. However, this leads to solutions that violate time and capacity constraints. This procedure is redone until the number of routes and the number of vehicles are equal.

The Procedure The improved FA is applied to CDVRPTW at the cluster level. A cluster is a group of customers who are geographically mapped based on the distance. Each vehicle is treated as a firefly, and it moves toward a brighter customer from a less bright one. This brightness depends on the distance between the customers, traffic data, and dynamic requests. The route is created by visiting customers until all customers have been visited. As detailed in Sect. 3.3 the sum of fixed costs and transport costs of all routes of the solution has been considered as the objective function. Therefore, altogether CDVRPTW is a minimization problem in which the fireflies with a lower value of objective function are the most attractive ones.

The comparison of distances between fireflies is made cluster by cluster. For example, two fireflies of cluster i composed of 10 customers can be represented as:

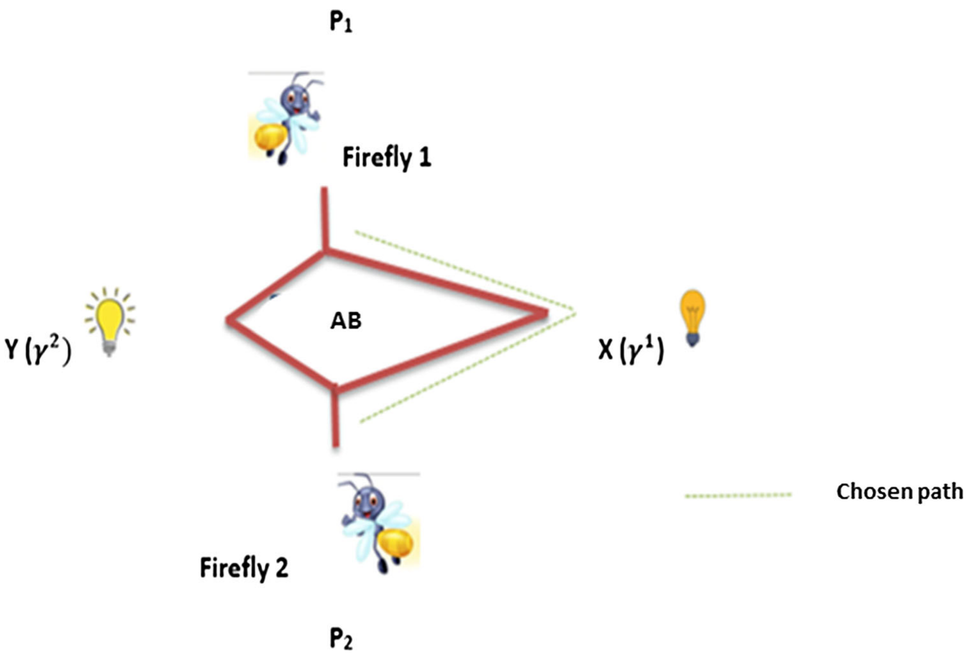


Fig. 4 The flashing process of fireflies under real-world scenarios

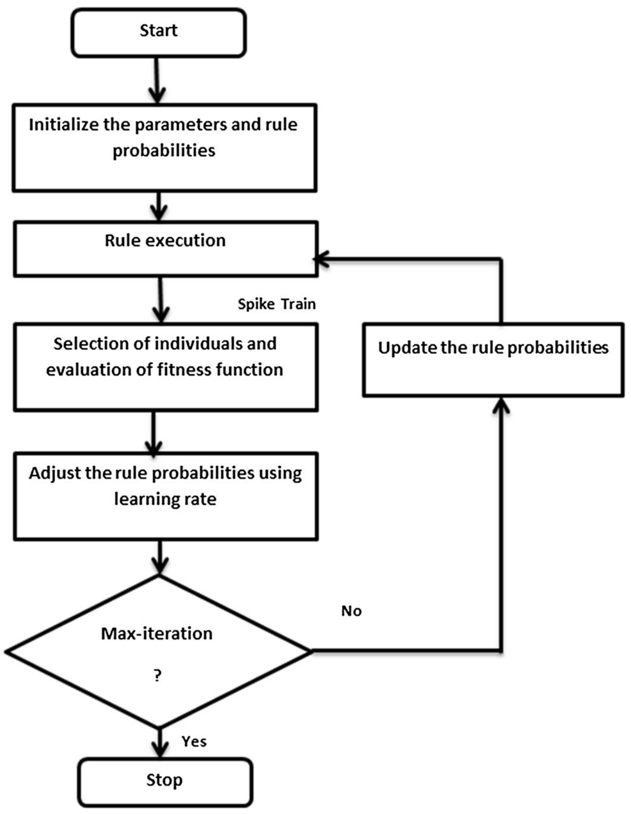

Start
Initialize the parameters and rule probabilities
Rule execution
Spike Train
Selection of individuals and evaluation of fitness function
Update the rule probabilities
Adjust the rule probabilities using learning rate
Max-iteration ?
No
Yes
Stop

Fig. 5 Flow diagram of SN *P* system parameter optimization

$$X_1(\text{cluster}_i) = \{0, 2, 3, 5, 7, 8, 1, 4, 6, 9\}$$
$$X_2(\text{cluster}_i) = \{2, 0, 5, 3, 7, 8, 1, 4, 6, 9\}$$

The hamming distance between X_1 and X_2 for the *i*th cluster is 4. This analysis is made for every cluster, and the total distance between two fireflies is the sum of all the distances for every cluster.

In addition, instead of focusing only on the best optimal solution, the first 3 optimal solutions are selected and are assigned to 3 different spiking neurons. The reason for selecting the first 3 optimal solutions can be stated as a measure of the balance between exploitation and control. The number of solutions selecting for next iterations should be mature enough to find the global optimal solutions in a reasonable time using the minimum number of spiking neurons. Choosing a higher number of solutions may lead to an increase in the complexity of controlling and coordinating neuronal communications. The remaining solutions are collected for random movement to get diverse solution space. So, in the successive iterations, the chances of getting trapped in local minimum can be avoided and the

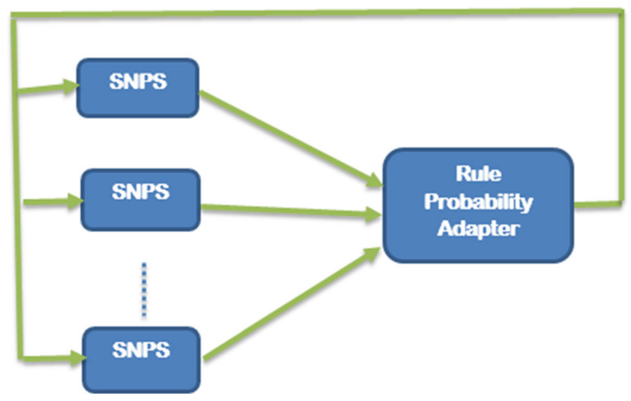


Fig. 6 A family of SN *P* systems connected through an adapter for rule selection

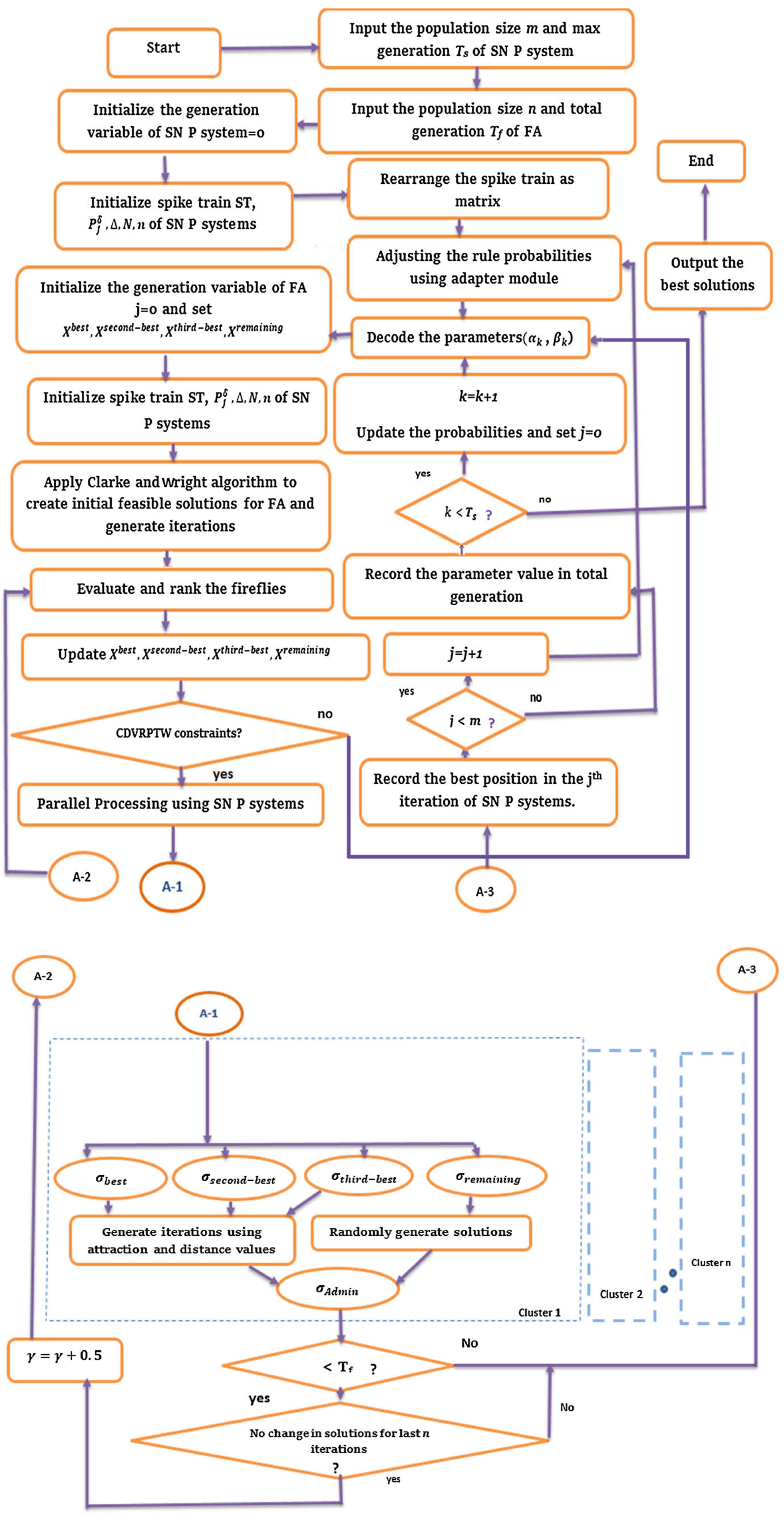

Start
Input the population size m and max generation T_s of SN P system
Input the population size n and total generation T_f of FA
Initialize the generation variable of SN P system=o
Initialize spike train ST, P_j^δ, Δ, N, n of SN P systems
Rearrange the spike train as matrix
Adjusting the rule probabilities using adapter module
Decode the parameters(α_k, β_k)
Initialize the generation variable of FA j=o and set X^{best}, $X^{second-best}$, $X^{third-best}$, $X^{remaining}$
Initialize spike train ST, P_j^δ, Δ, N, n of SN P systems
k=k+1
Update the probabilities and set j=o
Apply Clarke and Wright algorithm to create initial feasible solutions for FA and generate iterations
$k < T_s$?
yes
no
End
Output the best solutions
Evaluate and rank the fireflies
Record the parameter value in total generation
Update X^{best}, $X^{second-best}$, $X^{third-best}$, $X^{remaining}$
j=j+1
CDVRPTW constraints?
$j < m$?
Parallel Processing using SN P systems
Record the best position in the jth iteration of SN P systems.
A-2
A-1
A-3
σ_{best}
$\sigma_{second-best}$
$\sigma_{third-best}$
$\sigma_{remaining}$
Generate iterations using attraction and distance values
Randomly generate solutions
σ_{Admin}
Cluster 1
Cluster 2
Cluster n
$< T_f$?
No
$\gamma = \gamma + 0.5$
No change in solutions for last n iterations ?

Fig. 7 Graphical representation of IFA using SN P system optimization

optimal solutions can be generated more efficiently. It further improves the convergence rate due to the simultaneous exploration of solution space.

The initial value of parameter γ is set to 0.95. The value is chosen from several studies [89, 90] and will vary once the system enters into stagnation, and in that case, the value is reinitialized.

6 Spiking neural firefly optimization system (SFO) design

The proposed system is based on the flashing behavior of fireflies. A collection of different SN P systems is combined to produce the required spiking neural firefly optimization scheme (SFO). The SN P systems employed in the proposed scheme include arithmetic operator P systems, spike generator P systems, sorting P systems, random number generator P systems, objective function P systems, and improved firefly optimization P systems. The proposed SFO system is of the form:

$$\Pi_{\text{SFO}} = \left(\Pi_{\text{arith}}, \Pi_{sg}, \Pi_{rg}, \Pi_{\text{sort}}, \Pi_{\text{function}}, \Pi_{\text{IFA}}\right),$$

where Π_{arith} includes operations for addition, subtraction, multiplication, and division using SN P systems [91]. Π_{sg} and Π_{rg} are used for spike and random generators, respectively [92, 93]. Π_{function} indicates the SN P system for objective function evaluation based on the proposed scheme. Π_{sort} indicates the P system for sorting [94], and Π_{IFA} is used to denote the improved firefly algorithm using the SN P system for optimization as described in Sect. 5. Figure 8 shows the block diagram of the proposed SFO. Lines and arrows indicate the interactions between different modules in SFO. Lines are used for bidirectional communications, and arrows are used for unidirectional communications.

6.1 Design of Π_{IFA}

Definition 5 We define a Π_{IFA} of degree $m_{\text{IFA}} \geq 3$ of the form,

$$\Pi_{\text{IFA}} = (O, \sigma, syn, I_{in}, I_o)$$

1. O is the singleton alphabet called a spike.
2. σ is the set of neurons in the IFA system. $\sigma = \sigma_s \cup \sigma_{best} \cup \sigma_{second-best} \cup \sigma_{third-best} \cup \sigma_{remaining} \cup \sigma_{admin} \cup \sigma_{I_{in}} \cup \sigma_{I_o}$ where $\sigma = \sigma_1 \ldots \sigma_{m_{\text{IFA}}}$. I_{in},I_o indicate input and output neurons, respectively. σ_s is the mediator neuron. $\{\sigma_{best} \cup \sigma_{second-best} \cup \sigma_{third-best} \cup \sigma_{remaining} \cup \sigma_{admin}\}$ is referring to a firefly system where $\sigma_{best} \cup \sigma_{second-best} \cup \sigma_{third-best}$ is the best, second-best, and third-best neurons of the solutions. The σ_{admin} is a central neuron that contains the optimal solutions after each iteration. The Π_{IFA} starts with the initial configuration $Init_c$ and then enters into flashing configuration $Flash_c$ and halting with $Halt_c$.

   Every neuron in Π_{IFA} is of the form,

$$\sigma_{admin} = (n_l, R_l), 1 \leq l \leq m_{\text{IFA}},$$
$$\sigma_{best_i} = (n_i, R_i), 1 \leq i \leq m_{\text{IFA}},$$
$$\sigma_{second-best_j} = \left(n_j, R_j\right), 1 \leq j \leq m_{\text{IFA}},$$
$$\sigma_{third-best_k} = (n_k, R_k), 1 \leq k \leq m_{\text{IFA}},$$

   $\sigma_{I_{\text{in}}} = (n_{\text{in}}, R_{\text{in}})$ and $\sigma_{I_o} = (n_o, R_o)$

   σ_{admin} is responsible for keeping the values of firefly positions and updating the best, second-best, and third-best solutions. The number of spikes is indicated by n with the index denoting the corresponding neuron. Every firefly has five neurons out of which four neurons are used to indicate its levels (best, second-best, third-best, and remaining) and one admin neuron to track the updated values.
3. Synapses:

$$\begin{aligned} syn = \{&(\sigma_{admin}, \sigma_{admin}),\\ &(\sigma_{admin}, \sigma_{best}), (\sigma_{admin}, \sigma_{second-best}),\\ &(\sigma_{admin}, \sigma_{third-best}), \left(\sigma_{admin}, \sigma_{remaining}\right),\\ &(\sigma_{admin}, \sigma_{I_o}), (\sigma_{admin}, \sigma_{I_{in}}),\\ &\left(\sigma_{admin}, \Pi_{sg}\right), (\sigma_{admin}, \Pi_{sort})\\ &(\sigma_{admin}, \Pi_{arith}), \left(\sigma_{admin}, \Pi_{rg}\right),\\ &\left(\sigma_{admin}, \Pi_{function}\right), (\sigma_{admin}, \Pi_{IFA}),\\ &(\sigma_{best}, \Pi_{arith}), (\sigma_{second-best}, \Pi_{arith}),\\ &(\sigma_{third-best}, \Pi_{arith}), \left(\sigma_{remaining}, \Pi_{arith}\right)\} \end{aligned}$$

4. The input neuron I_{in} receives parameters of the algorithm to initialize the system. The parameters include the initial position of fireflies and the maximum number of iterations.
5. The output neurons I_o act as monitors by checking the optimal solutions and maximum iterations of the algorithm. It is also responsible for sending the best, the second-best, and the third-best values and activates forgetting rules to re-instantiate or terminate the whole system.

Each firefly is denoted by a set of $\{\{\sigma_{admin}, \sigma_{best}, \sigma_{second-best}, \sigma_{third-best} \text{and} \sigma_{remaining}\}$ neurons. The mediator neuron σ_s distributes position for all admin neurons σ_{admin} and further broadcasts values of the best, the second-best, the third-best solutions, and iteration number. The power of the signal [95] is employed to give the needed power to reach to all admin neurons.

The Admin Neuron (σ_{admin}) The admin neurons are associated with a set of rules to perform the steps of the

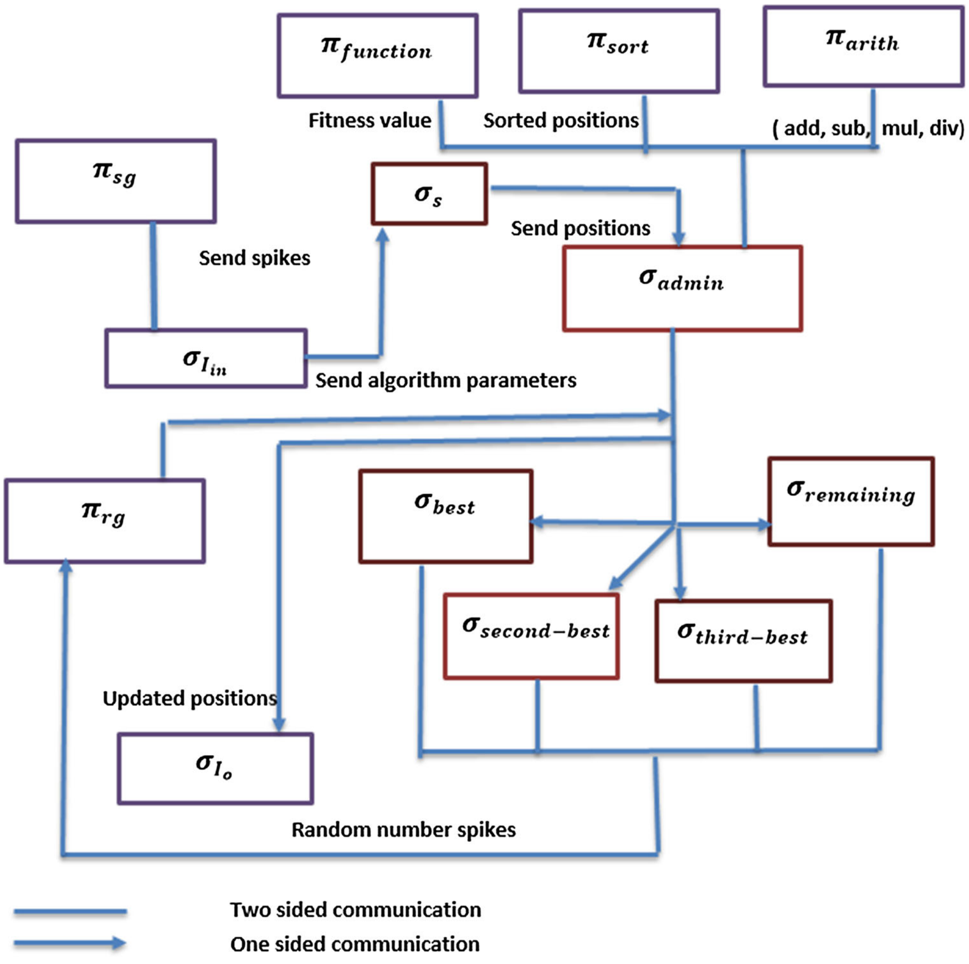


Fig. 8 Block diagram of the proposed SFO system

algorithm. Each admin neuron σ_{admin} has one initial spike and is loaded in the initial configuration $(Init_c)$.

The tasks of σ_{admin} are:

1. Accept algorithm variables and firefly positions to send to the best, second-best, third-best, and remaining neurons.
2. Calculate the values of the optimization equation and find feasible solutions.
3. Update the best optimal firefly in each stage.

All the rules are considered and controlled by time steps. This allows the incorporation of delay mechanisms in all rules, which is appropriate for various neuron stage activations. The admin neuron σ_{admin} starts receiving spikes from mediator neuron σ_s, and system changes its transition from $Init_c$ to $Flash_c$ state. Then it sends the positional information of fireflies from mediator neurons to $\sigma_{best}, \sigma_{second-best}, \sigma_{third-best},$ and $\sigma_{remaining}$ neurons. At the same time $\Pi_{function}$ is invoked by σ_{admin} to perform objective function evaluation. After this, it clears the neurons by activating the forgetting rules. After finishing the objective function evaluation, it invokes Π_{sort} in order to sort and rank the solutions, which are needed for the $\sigma_{best}, \sigma_{second-best}, \sigma_{third-best},$ and$\sigma_{remaining}$ neurons. All the synapses, which are no longer needed, are deleted, and the $\sigma_{best}, \sigma_{second-best}, \sigma_{third-best},$ and$\sigma_{remaining}$ neurons update the positions in the current iteration. After this, all admin neurons enter halt configuration $Halt_c$.

The $\sigma_{best}, \sigma_{second-best}, \sigma_{third-best},$ *and* $\sigma_{remaining}$

Neurons These neurons are used for calculating the firefly positions in 4 levels. The number of spikes for instantiating these neurons is two. The new positions are calculated and updated by sending encoded information in spikes to the respective neurons at fixed time steps. After each iteration, the spikes are cleared by forgetting rules. The structure of the neurons representing the fireflies at runtime is shown in Fig. 9.

The Input Neuron The input neuron is responsible for giving the algorithm parameters and constant values along with initial feasible solutions for the fireflies to start execution. After receiving the parameters, it changes the configuration from $Init_c$ to $Flash_c$. The respective neurons are updated with the iteration number, initial feasible solutions, and initial firefly positions. It can also receive incoming spikes from σ_{I_o} containing 3 optimal solutions and the remaining solutions.

The Mediator Neuron (σ_s) The functionalities of the mediator neurons are to act as mediators between the supporting SN *P* systems and Π_{IFA} and are provided with the power of signal strategy. The main function is to broadcast the values or objects to respective neurons to indicate the lifetime of spikes. The mediator neuron receives spikes from input neurons and distributes the parameters and positions to the admin neurons by controlling the synapses.

The Output Neuron (σ_{I_o}) The responsibilities of output neurons include:

1. Increment the iteration number by 1 each time the algorithm proceeds.
2. Halt when the current iteration value and the maximum iteration value are the same.
3. Send spikes of position vectors to each of the $\sigma_{best}, \sigma_{second-best}, \sigma_{third-best},$ and $\sigma_{remaining}$ neurons.
4. Send the final output to the surroundings.

6.2 Supporting SN *P* systems

The supporting SN *P* systems include SN *P* systems for random number generations, objective function evaluation, arithmetic operations, sorting, and spike generators. An SN *P* system for objective function evaluation is designed to adapt to the fireflies' flashing behavior based on the proposed scheme, while the remaining systems are found in the literature.

Definition 6 An SN *P* system for objective function evaluation (($\Pi_{function}$) of degree $m_{function} \geq 1$ can be defined by the tuple:

$$\Pi_{function} = \left(O, \sigma_i, s_n, I_{fin}, I_{fo}\right) \text{ where}$$

1. $O = \{a\}$ is the singleton alphabet called spike.
2. Set of neurons of the form,

   $$\sigma_i = (n_i, R_i), 1 \leq i \leq m_{function},$$

   where n_i are the initial spikes in neuron σ_i, which are employed according to the objective function. R_i is the rule set for calculating the new firefly positions and is encoded as $\varphi(a^P)$ which will be broadcast to the admin neurons.
3. $s_n = \{(I_{\text{fin}}, \Pi_{\text{IFA}}), \ldots, (I_{\text{fo}}, \Pi_{\text{IFA}})$
4. The input neuron I_{fin} receives spikes which are encoded by the initial position information of fireflies from Π_{IFA}
5. The I_{fo} is the output neuron which sends the fitness value encoded by $\varphi\left(a^{P.\text{value}}\right)$ to Π_{IFA}

The pseudocode of the proposed SFO system is given in algorithm 1. The algorithm starts by initializing the parameters according to the dataset. It uses the functionalities of the supporting SN *P* systems. The values are analyzed in each iteration to see whether the optimal

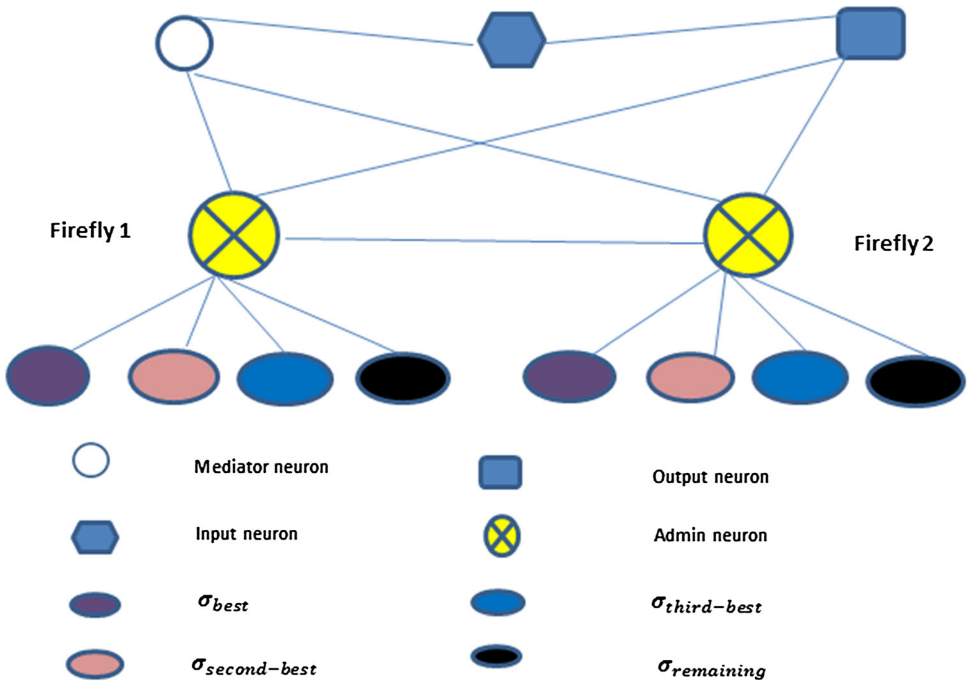


Fig. 9 Structure of neurons in Π_{IFA} for 2 fireflies

solutions are reached or not. The encoded result $\varphi(X_1)$ and $(X_1.\text{value})$, the best firefly is finally returned by the output neuron.

Algorithm 1. Pseudo-code of SFO system

```
Input: $\Pi=\{\sigma, I_{in}, I_o\}$, $maxIter$, $p_f$ and $maxTime$.
Output: $X_1$, $X_1.value$ and T
1.  $u \leftarrow 1, T \leftarrow 0$.
2.  Import the supporting SN P systems $\Pi_{arith}, \Pi_{sg}, \Pi_{rg}$ and $\Pi_{sort}$ .
3.  Invoke $\Pi_{arith}, \Pi_{sg}, \Pi_{rg}$ and $\Pi_{function}$
4.  While $u \leq maxIter$ and $T \leq maxTime$ do
5.    Invoke $\Pi_{sg}$
6.    Input $P_f$, $P_f.value$ and $a$ from $\sigma_{I_{in}}$ to $\sigma_s$
7.    Send $P_f$, $P_f.value$ and $a$ to $\sigma_{admin}$
8.    Invoke the rules of $\sigma_{admin}$ in parallel.
9.    Invoke $\Pi_{function}$ for objective function evaluation.
10.   Send optimal solutions to $\sigma_{best}$, $\sigma_{second-best}$, $\sigma_{third-best}$ and $\sigma_{remaining}$
11.   Obtain Optimal solutions to $\sigma_{admin}$ from $\sigma_{best}$, $\sigma_{second-best}$, $\sigma_{third-best}$ and $\sigma_{remaining}$ .
12.   Calculate $p_f^{new}$ by $\Pi_{arith}$ : activate $\sigma_{admin}$.
13.   Invoke $\Pi_{sort}$.
14.   Send $p_f^{new}$ to $\sigma_{I_o}$ in parallel.
15.   State $\varphi(X_1)$ and $\varphi(X_1.value)$ by $\sigma_{I_o}$
16. End while
17. Return $\varphi(X_1)$ and $\varphi(X_1.value)$ by $\sigma_{I_o}$
```

7 Experimental results and discussion

In this section, the performance of the proposed SFO system to solve CDVRPTW will be rigorously assessed. As far as the authors are concerned, there are no benchmark datasets on CDVRPTW till date. So the SFO is tested on DVRP instances and CVRPTW instances separately, and the results are analyzed. The various parameter settings are given in Table 1. All the experiments are simulated using MATLAB with Intel xenon 2.93 GHz processor, 12 GB RAM, and Windows 10 OS.

7.1 Experiment 1: DVRP datasets

SFO is tested by an average of 40 runs on the dataset, which are open and available at http://neo.lcc.uma.es/vrp/ (dataset 1). A comparison of solution quality in terms of best and average values is compared with ACO, k-ACO, E-ACO [96], VNS [97], and GA-DVRP [98], and the results are analyzed.

Also, the utilization rate of a vehicle is an important measure for verifying the performance of the proposed approach. It estimates whether each of the vehicles used fully or not. It is analyzed in Fig. 10. From the figure, it can be noted that the utilization rates of vehicles are high in SFO (13 out of 20 instances) scheme. In the other 7 instances, SFO is inferior to E-ACO but superior to GA, ACO, and VNS. In addition, the best and average values of the solution quality of six approaches can be analyzed from Table 2. The best results are in boldface. The proposed SFO achieves 17 out of 20 best values and 15 out of 20 average values compared with other best-known schemes. The K-ACO attains the worst performance compared to the other three algorithms. In order to clearly differentiate these schemes, K-ACO is chosen as a benchmark to plot how the improvement of the remaining algorithms is compared with K-ACO. A boxplot is given for analyzing the improvement percentage of various schemes. The minimum, the maximum, the sample median, and the first and third quartiles are shown in Fig. 11. From the analysis of boxplots, it is noted that the improvement percentage of SFO is superior to other schemes in terms of solution quality.

From the analysis of all of the above results, we concluded that the proposed SFO system attains substantially better performance compared to other existing best-known schemes.

7.2 Experiment 2: CVRPTW datasets

The proposed SFO is applied over Solomon's benchmark instances (dataset 2), which are partitioned into 6 categories: C1, C2, R1, R2, RC1, and RC2. The time window at depot is restricted for R1, C1, and RC1 so that only limited customers can be accommodated. But, for R2, C2,

Table 1 Parameter settings

Parameter	Value
α	0.2
β	1
γ (initial value)	.95
MaxIter	100–300
Population size	30
H	60

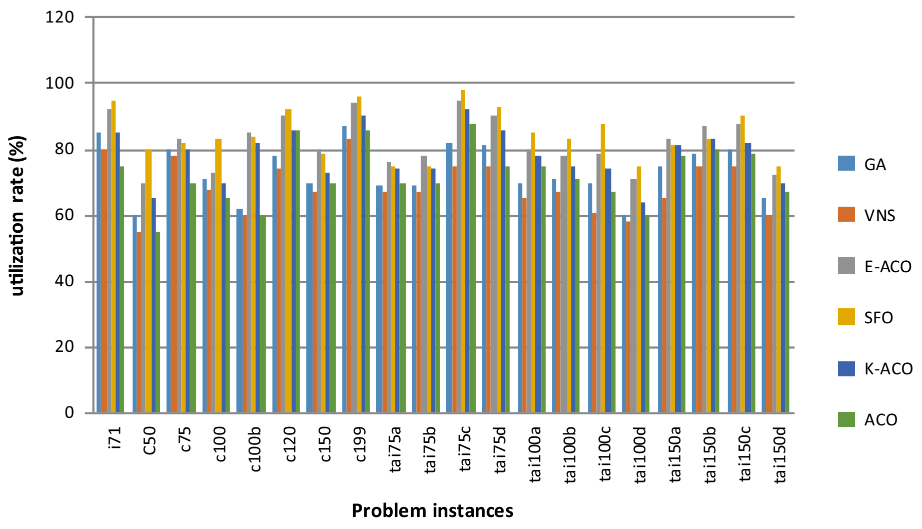


Fig. 10 A comparison of the utilization rate of various schemes for dataset 1

Table 2 Comparison of solution quality in terms of best and average values of various schemes (dataset 1)

Problem	ACO		K-ACO		E-ACO		VNS		GA-DVRP		SFO	
	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average	Best	Average
i71	311.18	358.69	286.42	315	259.71	**297.08**	304.32	325.18	288.3	319.49	**240.1**	312.8
C50	631.3	681.86	611.27	652.45	607.21	647.21	599.53	653.84	**566.01**	597.34	689.7	**595.8**
c75	1009	1042	982.30	1136.90	924.71	1045.44	981.64	1040.00	944.46	**990.78**	**911.5**	997.1
c100	973.26	1066.16	1021.00	1162	973	1044.96	1022.92	1087.18	**943.89**	988.15	1089	**919.8**
c100b	944.23	1023.60	921.41	1000.80	869.22	950.17	866.71	942.81	869.41	904.03	**768.9**	**795.7**
c120	1416.45	1525.15	1137.50	1297	1108.15	1197.68	1285.21	1469.24	1288.66	1399.40	**1100**	**1176.5**
c150	1345.73	145550	1480.00	1730.90	1378.63	1472.40	1334.73	1441.37	1273.50	**1359.25**	**1271.4**	1472
c199	1771.04	1844.82	1732.00	1971	1561.1	1836.9	1679.65	1769.95	1646.36	1700.54	**1532.7**	**1683.2**
tai75a	1841	1945.2	1657.10	2083.30	1690.91	1983.92	1806.81	1954.25	1744.78	1823.71	**1483.9**	**1593.1**
tai75b	1535.43	1704.06	1541.90	1680.70	1509.6	1647.78	2250.5	2462.50	2181.31	2290.95	**1324.6**	**1502.3**
tai75c	1574.98	1653.58	1395.10	1591.60	1329.42	1470.60	3479.44	3680.35	3280.79	3449.32	**1318.9**	**1412.8**
tai75d	1472.35	**1529.00**	1494	1663.10	1409	1661.73	1480.7	1560.71	1441.35	1546.18	**1322.4**	1795.2
tai100a	2375.92	2428.4	2497.6	2714.20	2281.7	2550.61	2169.1	2319.72	**2119.03**	2212.58	2281.6	**2315.9**
tai100b	2283.97	2347.90	2365.80	2624.70	2255.83	2500.72	2934.86	3089.57	2885.94	3073.58	**2134.8**	**2219.8**
tai100c	1562	1655.91	1548	1764.50	1442.45	1743.1	1621.03	1746.07	1433.73	1502.56	**1326.9**	**1519.8**
tai100d	2008.13	2061	1958.80	2228	1581	1844	1490.58	1557.81	1504.63	1589.76	**1498.2**	**1755.6**
tai150a	3644.78	3840.18	3800.60	4216.40	3307.63	3684.03	2674.29	2928.77	2593.78	**2759.96**	**2489**	5187.1
tai150b	3166.88	3327.47	3270.2	3654.80	3128.00	3439.38	1446.50	1541.98	1408.48	1434.56	**1315.9**	**3212**
tai150c	2311.48	3016	2795.00	2886.20	2583.4	2729.15	1969.94	2100.38	1793.64	1916.03	**1589.7**	**2615.1**
tai150d	3058.81	3203.75	2981.20	3245.21	2809	3186.1	2954.54	3147.88	2911.47	3010.34	**2698.7**	**3095.4**
average	1787	1885.5	1856.83	2064.07	1650	1846.64	1717.65	2540.06	1655.976	1743.426	**1577.385**	**1808.85**
Count	0	1	1	0	0	1	0	0	3	3	17	15

The best results are bolded

and RC2 the time window is broad with the goal that numerous clients can be overhauled in a similar route. The results of the proposed system are analyzed with the existing CVRPTW schemes, ant colony optimization (ACO) [99], ACO with tabu search [100], hybrid multiobjective evolutionary algorithm (HMOEA) [101], tissue P system with MOEA (PDVA) [102], and multiobjective goal programming (MOGP) [103]. A comparison is made on the vehicle utilization rate and solution quality and is given in Fig. 12 and Table 3, respectively. SFO

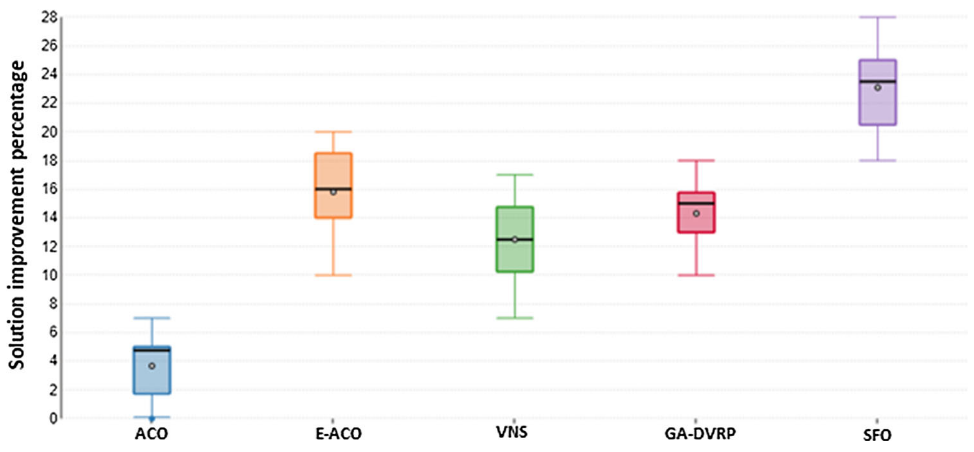


Fig. 11 Comparison of solution improvement percentage of various algorithms for dataset 1 (K-ACO as the benchmark scheme)

achieved 5 out of 6 of the best solution values over 6 algorithms. From the analysis of results, it is noted that the vehicle utilization rate of SFO is superior to other best-known schemes. Also, the solution improvement percentage is calculated as ACO tabu as a benchmark scheme and the performance of various algorithms is plotted using boxplots in Fig. 13. From the boxplot, it is concluded that SFO is having the best solution improvement percentage and ACO is having the worst improvement over other schemes.

Theorem 1 *The time complexity of SFO is* $(r \times T(IFA) + T\ (SS) + T\ (I_{\text{in}}) + T\ (I_o)) \times maxIter$ *where r is a constant used to weight the term which can be either 1 (if* $T(IFA)$ *alone is considered) or 2 (if* σ_{admin} *and* $T(IFA)$ *are considered) and* $T(IFA)$ *is the time units of firefly neurons.* $T\ (SS)$ *denotes the total time needed for all the supporting* SN *P systems.* $T(I_{in})$ *and* $T\ (I_o)$ *refer to the time steps needed for the input and output neurons, respectively, and maxIter is the maximum number of iterations of the algorithm.*

Proof Let the SFO system have n fireflies, each of which involves $5n$ neurons in total. The solution neurons run in parallel and operate sequentially with their admin neurons. Therefore, the needed time is T(IFA). The n firefly operations depend on T (SS). The maximal parallelism in SN P systems allows polynomial time complexity in solving optimization problems according to the number of supporting systems. Therefore, it is concluded that the total time complexity of SFO is purely dependent on the *maxIter* and T(SS). □

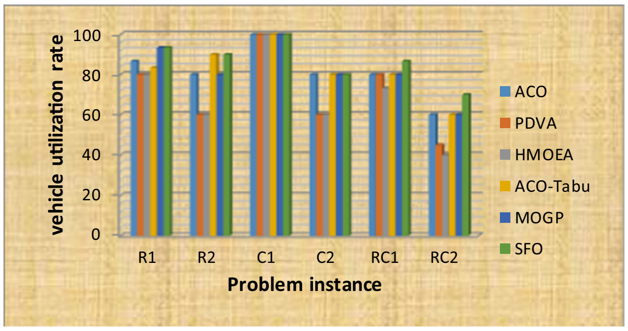


Fig. 12 Comparison of vehicle utilization rates of various schemes for dataset 2

The results from Theorem 1, Tables 2, and 3, Figs. 10, 11, 12, and 13 indicate the following conclusions:

1. The proposed SFO is superior to the best-known schemes in terms of solution quality and vehicle utilization rate.
2. With respect to the stopping conditions, the better balance between exploration and exploitation has stronger optimization capacity in polynomial time (Theorem 1).
3. The solution improvement percentage of SFO is higher compared to the best-known schemes.

7.3 Statistical analysis

Based on the experimental outcomes, a set of statistical analysis of algorithms over the instances has been made. We adopted a parametric test, pairwise t test to check the significance of SFO over other schemes and nonparametric test, Wilcoxon's, and Friedman's test to see whether the outcomes of the algorithms are significantly different or not regardless of the relationship between them. In addition, a Holm–Bonferroni [104] ranking scheme is adopted on the basis of the average solution quality obtained over 20 test instances for dataset 1 and 6 problem categories for dataset 2. All the experiments are conducted on IBM SPSS Statistics software with a significance level of 0.05.

Table 3 Comparison of solution values for various schemes (dataset 2)

	*R*1	*R*2	*C*1	*C*2	*RC*1	*RC*2
ACO	1383.2	1098.2	881.44	641.25	1211.12	1209.44
PDVA	1228.6	1033.53	828.38	591.49	1362.09	1068.26
HMOEA	1187.35	951.74	828.74	590.69	1355.36	1068.26
ACO tabu	1213.16	952.3	841.92	612.75	1415.62	1120.37
MOGP	1258.89	994.99	828.94	591.49	1384.3	1157.41
SFO	**1076.65**	**912.09**	832.13	**453.7**	**1185.32**	**996.25**

The best results are bolded

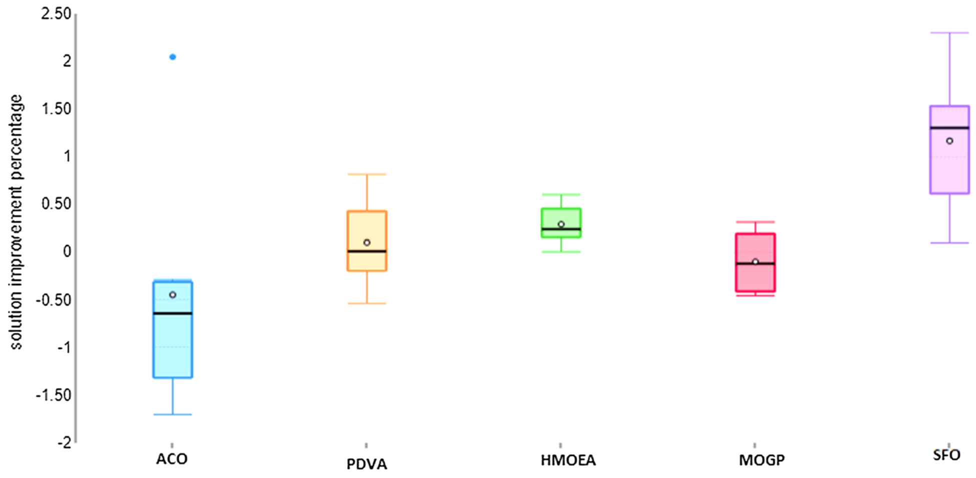


Fig. 13 Comparison of solution improvement percentage of various schemes for dataset 2 (ACO tabu as the benchmark scheme)

7.3.1 Pairwise *t* test

The proposed SFO is compared with the state-of-the-art algorithms using the t test and is tabulated in Tables 4 and 5, respectively, for dataset 1 and dataset 2. The null hypothesis for the paired t test is assumed, as the true mean difference between paired samples is zero. On the other hand, the alternative hypothesis is that the true mean difference between paired samples is not zero. The significance level is set as 0.05. Since all the pairs have significance < 0.05, it leads to the rejection of the null hypothesis. So there is a pairwise difference of means among the samples.

7.3.2 Wilcoxon's and Friedman's test

These methods are used to analyze the pairwise differences among the algorithms considered for experimentation. For Wilcoxon's test, the hypothesis is assumed as follows:

H_0 The medians of the pairs are identical
H_1 The medians for the pairs are not identical

The result of Wilcoxon's test (WT) for dataset 1 is given in Table 6. For dataset 2, the test is not as significant as the number of samples is less than 10.

For the pairs ACO and SFO, the value of $z = -2.9493$. The corresponding p value is 0.00318. In addition, the value of w is 26. The critical value for w at $n = 20$ is 43. So the result is significant which leads to the rejection of the null hypothesis.

For the pairs K-ACO and SFO, the value of $z = -3.5839$. The corresponding p value is 0.01209. In

Table 4 *T* test: paired samples statistics for dataset 1

	Mean	*N*	SD	SE mean	Sig
Pair 1					
ACO	1761.8915	20	857.2475	191.6864	.003
SFO	1419.395	20	605.8148	135.4643	
Pair 2					
E-ACO	1650.4835	20	831.3698	185.8999	.000
SFO	1419.395	20	605.8148	135.4643	
Pair 3					
K-ACO	1984.8967	20	975.99005	212.97849	.000
SFO	1419.395	20	605.8148	135.46	
Pair 4					
GA-VRP	1655.976	20	803.84	107.025	0.0395
SFO	1419.395	20	605.8148	135.46	0.0173
Pair 5					
VNS	1717.6410	20	830.7174	185.75	
SFO	1419.395	20	605.8148	135.46	0.0173

Table 5 *T* test: paired samples statistics for dataset 2

	Mean	*N*	SD	SE	Sig.
Pair 1					
ACO	1070.775	6	267.4418	109.1827	0.0135
SFO	909.35		255.0638	104.1294	
Pair 2					
PDVA	909.3567	6	276.9046	113.0458	.0095
SFO	1018.725		255.0638	104.1294	
Pair 3					
HMOEA	997.0233	6	270.2520	110.3299	0.0204
SFO	909.3567		255.0638	104.1294	
Pair 4					
ACO tabu	1026.0483	6	284.8475	116.2885	0.0164
SFO	909.3567		255.0638	104.1294	
Pair 5					
MOGP	1036.0033	6	292.5331	119.4261	.0092
SFO	909.3567		255.0638	104.1294	

addition, the value of w is 9. So the result is significant which leads to the rejection of the null hypothesis.

For the pairs EACO and SFO, the value of z is − 3.1359. The *p* value is .00168. In addition, the value of w is 21. The critical value of w at $N = 20$ is 23, which leads to the rejection of H_0. The result is significant.

For the pairs VNS and SFO, the value of z is − 3.2106; the *p* value is .00132. The result is significant at $p < 0.05$. Also, the value of w is 19. The critical value at $N = 20$ is 43 which leads to the rejection of the null hypothesis.

For the pairs GADVRP and SFO, the value of z is − 3.2106. The *p* value is .00132. The result is significant at $p < 0.05$. In addition, the value of w is 19. The critical value at $N = 20$ is 43 which leads to the rejection of the null hypothesis.

The Friedman test (FT) is used to evaluate the differences between the number of samples. It relies on the rank ordering of data rather than means and variances. The value of Friedman's test statistics for dataset 1 and dataset 2 is shown in Tables 7 and 8, respectively. In both cases, the *p* value is very small and is < 0.05. It indicates if the different treatments really are identical, what is the chance that random sampling would result in the sum of ranks as far apart as observed in the experiment. If *p* is small, reject the idea that all of the differences between columns are due to random sampling. Since the *p* value for both the datasets are small, the result is significant.

7.3.3 Holm–Bonferroni ranking

After analyzing the results so far, the algorithms are ranked on the basis of the solution quality for both datasets. Based on the ranks, SFO is taken as a reference scheme and the corresponding cumulative normal distribution values are calculated. It is then compared with the *D*/*j* values where $j = 1,2,\ldots5$ (5 schemes except SFO) with $D = 0.05$. The results are given in Table 9 and 10. The significance is given as 'Accepted' if the null hypothesis (no difference in performance between the considered algorithms) is true otherwise 'Rejected' if the null hypothesis is false.

For dataset 2, it is noted that the performance of HMOEA and PDVA has the same performance as that of SFO. Therefore, it is concluded that the membrane-based algorithms and evolutionary schemes have similar performance to that of SFO compared to other schemes.

The experimental outcomes from Tables 4, 5, 6, 7, 8, 9, and 10 reveal the following conclusions:

1. The *t* test statistics show that SFO is really better than other schemes since there are significant differences between the solutions for both datasets.
2. From the calculated *p* values of WT and FT over the pairs, it is noted that the SFO is statistically superior to other schemes.
3. The Holm–Bonferroni ranking shows that the SFO is the highest in ranking and is able to outperform all nonmembrane-based schemes.

The results summarized above appear competitive with respect to state-of-the-art algorithms.

Table 6 WT test analysis for dataset 1

	W value	Mean difference	Sum of positive ranks	Sum of negative ranks	*Z* value	Mean	SD
ACO and SFO	26	16,634.05	184	26	− 2.9493	105	26.79
K-ACO and SFO	9	1084.16	201	9	− 3.5839	105	26.79
E-ACO and SFO	21	960.78	189	21	− 3.1359	105	26.79
VNS and SFO	19	1027.95	191	19	− 3.2106	105	26.79
GA-VRP and SFO	36	966.28	174	36	− 2.576	105	26.79

Table 7 FT summary for dataset 1

Calculation summary
$X_r^2 = 0.02 * 19{,}564–360$
$X_r^2 = 31.28$
The X_r^2 statistic is 31.28 (4, $N = 20$)
The p value is $< .00001$
The result is significant at $p < .05$

Table 8 FT summary for dataset 2

Calculation summary
$X_r^2 = 17.668$
The X_r^2 statistic is 17.668 (5, $N = 6$)
The p value is < 0.003392
The result is significant at $p < .05$

Table 9 Holm–Bonferroni ranking for dataset 1 (reference scheme-SFO (rank 5.15))

	RANK	ZJ	PJ	*D*/*J*	Output
E-ACO	4.85	− 2.08333	0.01401	0.0185	Rejected
K-ACO	4	− 3.43137	0.007143	0.025	Rejected
GA-DVRP	3	− 5.26961	< 0.00001	0.05	Rejected
VNS	2.8	− 5.4321	< 0.00001	0.024	Rejected
ACO	2	− 5.1098	< 0.00001	0.0328	Rejected

Table 10 Holm–Bonferroni ranking for dataset 2 (reference scheme-SFO (rank 5.5))

	RANK	ZJ	PJ	D/J	output
HMOEA	4.833	− 0.617	0.268	0.05	Accepted
PDVA	3.833	− 1.54306	0.0614	0.025	Accepted
ACO tabu	3	− 2.3145	0.008	0.0166	Rejected
MOGP	3	− 2.3145	0.008	0.0166	Rejected
ACO	2	− 3.24037	0.00059	0.01	Rejected

7.3.4 Post Hoc procedures

The main drawback of the FT is that it can only detect significant differences over multiple comparisons, being not able to define proper differences among the schemes. Therefore, a post hoc test [105] is applied to obtain a *p* value, which defines the degree of rejection of each hypothesis. We adopted Conover method with 2 types of *p* value adjustment, namely Holm FWER (forward error rate) and Benjamini–Hochberg FDR method [106]. These tests are used to discern which of the sample pair combinations are significantly different. Both FWER and FDR methods are used for reducing type 1 error inflation that leads to false-positive discovery rate. The FWER and FDR adjustment for dataset 1 is, respectively, indicated in Figs. 14 and 15.

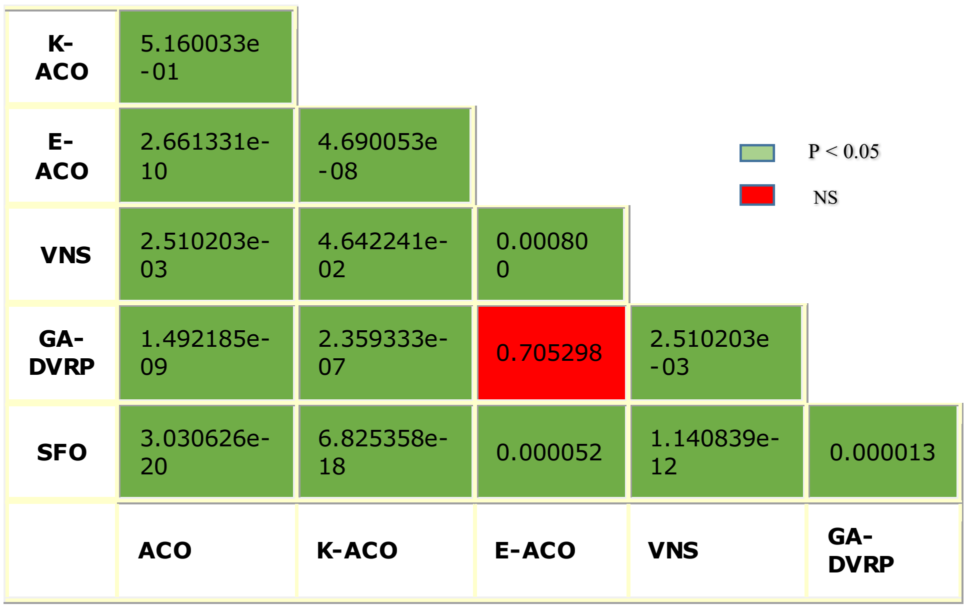
Post hoc *p* values of all possible pairs (of samples/groups) can be represented as a lower triangular matrix. Each entry is the *p* value of row/column pair, i.e., the null hypothesis that the group represented by a particular column name is different from the group by a particular row name. The pair, which is significant, is shaded using green color. The nonsignificant pairs are colored using red.

From Figs. 14 and 15, it is indicated that the pair GA-DVRP and E-ACO pair is not significant. All the pairs involving SFO are significant. For dataset 2, there are 4 pairs that are significant in the FWER method and are indicated in Fig. 16. Among the 4 pairs 3 pairs involving SFO gives a significant result. In the FDR method, there are a total of 8 pairs which are significant. Among those, 4 pairs are having SFO which are significant. So finally, it can be concluded that the significance of SFO is higher compared to other schemes in both FWER and FWR methods of post hoc analysis (Fig. 17).

8 Conclusions and future scope

This article suggests an efficient SN *P* system combined with firefly optimization architecture for addressing the problems of vehicle routing, namely CDVRPTW. In this scheme, we suggested a feasible way of using the SN *P* system to develop an optimization method for having the approximate solutions of the problem instance. It uses Clarke and Wright algorithm to define initial assignments of fireflies to find solutions to CDVRPTW. A high degree of parallelism is employed by carefully designing the firefly–neuron combinations, and the parameters in the firefly scheme are well optimized by adjusting the rule probabilities of SN *P* systems. The authors presented the algorithms, complexity analysis, and experimental results with statistical validation to check the effectiveness of the algorithm. This study being the first attempt in this kind, the results appear to be positive and successful relative to ad hoc optimization algorithms. The experimental outcomes justify the novelty, effectiveness, and feasibility of the system.

The main strengths of the proposed system include:

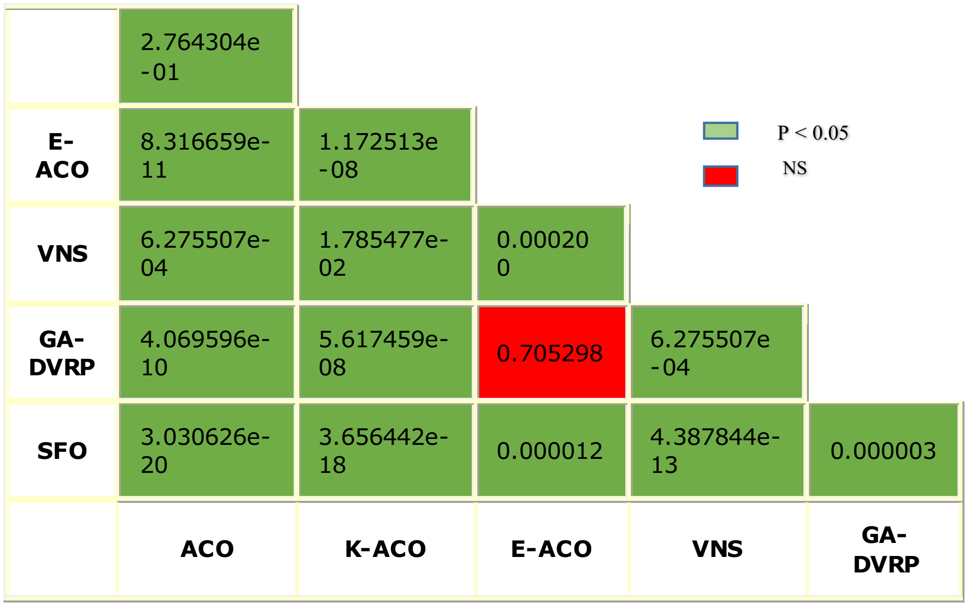


K-ACO	5.160033e-01				
E-ACO	2.661331e-10	4.690053e-08			
VNS	2.510203e-03	4.642241e-02	0.000800		
GA-DVRP	1.492185e-09	2.359333e-07	0.705298	2.510203e-03	
SFO	3.030626e-20	6.825358e-18	0.000052	1.140839e-12	0.000013
	ACO	K-ACO	E-ACO	VNS	GA-DVRP

🟩 P < 0.05

🟥 NS

Fig. 14 Conover *p* values, further adjusted by the Holm FWER method (dataset 1)



	2.764304e-01				
E-ACO	8.316659e-11	1.172513e-08			
VNS	6.275507e-04	1.785477e-02	0.000200		
GA-DVRP	4.069596e-10	5.617459e-08	0.705298	6.275507e-04	
SFO	3.030626e-20	3.656442e-18	0.000012	4.387844e-13	0.000003
	ACO	K-ACO	E-ACO	VNS	GA-DVRP

🟩 P < 0.05

🟥 NS

Fig. 15 Conover *p* values, further adjusted by the Benjamini–Hochberg FDR method (dataset 1)

1. The spiking neural P systems are operating in parallel on neuron level and sequential on system level. This point underlines the difference of earlier studies with the proposed scheme.
2. The optimization procedure is done by a spiking neural P system not by a human-designed algorithm.
3. The idea can be used for other applications such as fault diagnosis of power stations, path planning of robots, unmanned aerial vehicle routing, image segmentation.

The drawbacks of the proposed scheme are:

1. The time complexity depends on the iterations and supporting SN *P* systems. So for applications which need more number of iterations and supporting systems, the complexity will be high.
2. The power of signal strategy needs to be carefully designed for each neuronal communications.

For future works, the authors will try to overcome the afore-mentioned drawbacks.

Other research dimensions can be stated as follows:

1. Expand the method to handle more complex real-world dynamic instances.

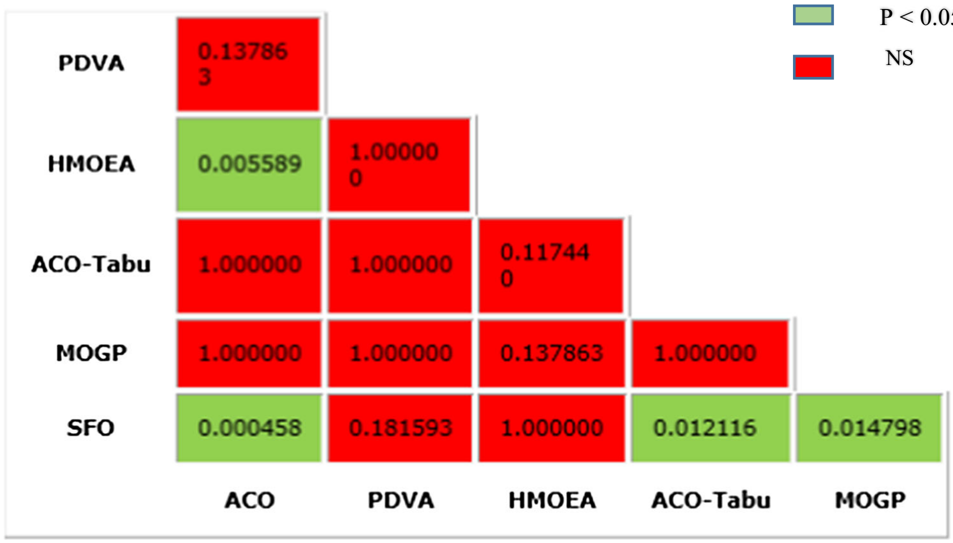


Fig. 16 Conover p values, further adjusted by the Holm FWER method (dataset 2)

	ACO	PDVA	HMOEA	ACO-Tabu	MOGP
PDVA	0.137863				
HMOEA	0.005589	1.000000			
ACO-Tabu	1.000000	1.000000	0.117440		
MOGP	1.000000	1.000000	0.137863	1.000000	
SFO	0.000458	0.181593	1.000000	0.012116	0.014798

🟩 P < 0.05
🟥 NS

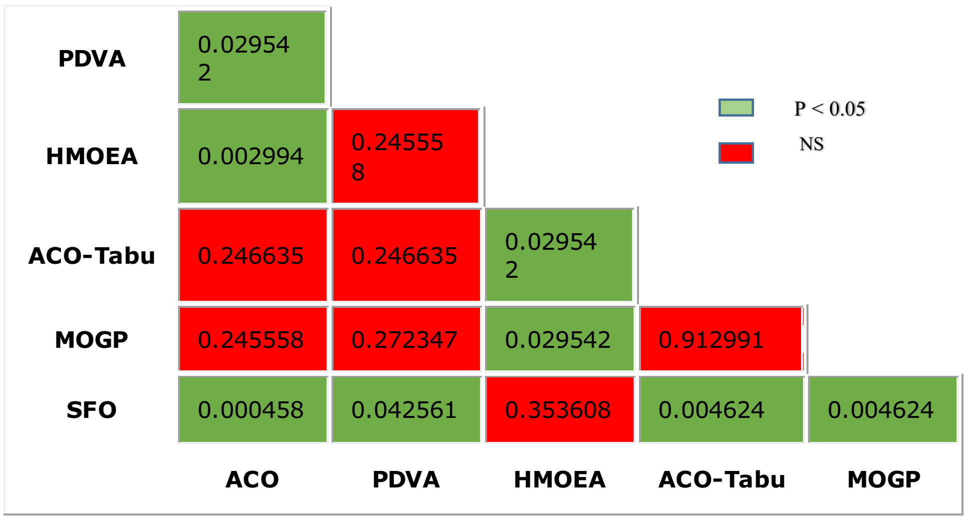


Fig. 17 Conover p values, further adjusted by the Benjamini–Hochberg FDR method (dataset 2)

	ACO	PDVA	HMOEA	ACO-Tabu	MOGP
PDVA	0.029542				
HMOEA	0.002994	0.245558			
ACO-Tabu	0.246635	0.246635	0.029542		
MOGP	0.245558	0.272347	0.029542	0.912991	
SFO	0.000458	0.042561	0.353608	0.004624	0.004624

🟩 P < 0.05
🟥 NS

2. Improve the convergence rate further by carefully designing the modules.
3. Consider timed SN P systems to control and coordinate the whole subsystems

Funding This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Rozenberg G, Bäck T, Kok JN (eds) (2012) Handbook of natural computing. Springer, Berlin, pp 461–477
2. Agatonovic-Kustrin S, Beresford R (2000) Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. J Pharm Biomed Anal 22(5):717–727. https://doi.org/10.1016/S0731-7085(99)00272-1
3. Zhang Y, Agarwal P, Bhatnagar V, Balochian S, Yan J (2013) Swarm intelligence and its applications. Sci World J. https://doi.org/10.1155/2013/528069
4. Bartz-Beielstein T, Branke J, Mehnen J, Mersmann O (2014) Evolutionary algorithms. Wiley Interdiscip Rev Data Min Knowl Discov. https://doi.org/10.1002/widm.1124
5. Gruska J (1999) Quantum computing, vol 2005. McGraw-Hill, London
6. Păun G (2010) A quick introduction to membrane computing. J Logic Algebr Program 79(6):291–294. https://doi.org/10.1016/j.jlap.2010.04.002
7. Păun G, Rozenberg G (2002) A guide to membrane computing. Theor Comput Sci 287(73):100–3975. https://doi.org/10.1016/S0304-3975(02)00136-6

8. Zhao Y, Liu X, Wang W (2016) Spiking neural P systems with neuron division and dissolution. PLoS ONE 11:e0162882. https://doi.org/10.1371/journal.pone.0162882
9. Pan L, Păun G, Pérez-Jiménez M (2011) Spiking neural P systems with neuron division and budding. Science China. Inf Sci 54:1596–1607. https://doi.org/10.1007/s11432-011-4303-y
10. Song X, Wang J, Peng H, Ning G, Sun Z, Wang T, Yang F (2018) Spiking neural P systems with multiple channels and anti-spikes. Biosystems 169–170:13–19. https://doi.org/10.1016/j.biosystems.2018.05.004
11. Chen H, Freund R, Ionescu M, Paun G, Pérez-Jiménez M (2007) On string languages generated by spiking neural P systems. Fundam Inform 75:141–162
12. Zhang G, Rong H, Neri F, Pérez-Jiménez M (2014) An optimization spiking neural P system for approximately solving combinatorial optimization problems. Int J Neural Syst 24:1440006. https://doi.org/10.1142/S0129065714400061
13. García-Arnau M, Pérez D, Rodríguez-Patón A, Sosík P (2009) Spiking neural P systems: stronger normal forms. IJUC 5:411–425
14. Pérez-Jiménez MJ (2010) A computational complexity theory in membrane computing. In: WMC 2009. Lecture notes in computer science, vol 5957. Springer, Berlin. https://doi.org/10.1007/978-3-642-11467-0_10
15. Chen Z, Zhang P, Wang X, Shi X, Wu T, Zheng P (2016) A computational approach for nuclear export signals identification using spiking neural P systems. Neural Comput Appl. https://doi.org/10.1007/s00521-016-2489-z
16. Díaz-Pernil D, Gutiérrez-Naranjo M (2017) Semantics of deductive databases with spiking neural P systems. Neurocomputing. https://doi.org/10.1016/j.neucom.2017.07.007
17. Diaz C, Frias T, Sanchez G, Perez-Meana H, Toscano K, Duchen G (2017) A novel parallel multiplier using spiking neural P systems with dendritic delays. Neurocomputing. https://doi.org/10.1016/j.neucom.2017.02.009
18. Tingfang W, Wang Y, Jiang S, Yansen S, Shi X (2018) Spiking neural P systems with rules on synapses and anti-spikes. Theor Comput Sci 724:13–27. https://doi.org/10.1016/j.tcs.2017.12.015
19. Deb K (2014) Multi-objective optimization. In: Search methodologies. Springer, Boston, pp 403–449
20. Bansal JC, Sharma H, Jadon SS et al (2014) Spider monkey optimization algorithm for numerical optimization. Memet Comput 6:31. https://doi.org/10.1007/s12293-013-0128-0
21. Brownlee J (2011) Clever algorithms: nature-inspired programming recipes, Lulu.com
22. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
23. Zmazek B, Taranenko A, Smid M (2005) Capacitated VRP with time windows and multiple trips within working day, pp 104–109. https://doi.org/10.1109/iti.2005.1491105
24. Cardoso Pedro JS, Schütz G, Mazayev A, Ey E, Corrêa T (2015) A Solution for a Real-time Stochastic Capacitated Vehicle Routing Problem with Time Windows. Procedia Computer Science 51:2227–2236. https://doi.org/10.1016/j.procs.2015.05.501
25. Yang XS (2009) Firefly algorithms for multimodal optimization. In: Watanabe O, Zeugmann T (eds) Stochastic algorithms: foundations and applications. SAGA 2009. Lecture notes in computer science, vol 5792. Springer, Berlin
26. Gandomi A, Yang X, Alavi A (2011) Mixed variable structural optimization using firefly algorithm. Comput Struct 89(23):2325–2336
27. Gao M, He X, Luo D, Jiang J, Teng Q (2013) Object tracking using firefly algorithm. IET Comput Vis 7(4):227–237
28. Altabeeb AM, Mohsen AM, Ghallab A (2019) An improved hybrid firefly algorithm for capacitated vehicle routing problem. Appl Soft Comput 84:105728. https://doi.org/10.1016/j.asoc.2019.105728
29. Osaba E, Yang X, Diaz F et al (2017) A discrete firefly algorithm to solve a rich vehicle routing problem modelling a newspaper distribution system with recycling policy. Soft Comput 21:5295–5308. https://doi.org/10.1007/s00500-016-2114-1
30. Aggarwal D, Chahar V, Girdhar A (2017) Firefly algorithm for the vehicle routing problem with time windows. https://doi.org/10.1109/icacci.2018.8554555
31. Yesodha R, Amudha T (2019) An improved firefly algorithm for capacitated vehicle routing optimization. In: 2019 amity international conference on artificial intelligence (AICAI), Dubai, United Arab Emirates, pp 163–169. https://doi.org/10.1109/aicai.2019.8701269
32. Clarke G, Wright JW (1964) Scheduling of vehicles from a central depot to a number of delivery points. Oper Res 12(4):568–581
33. Crainic TG, Gendreau M, Potvin J-Y (2009) Intelligent freight-transportation systems: assessment and the contribution of operations research. Transp Res Part C Emerg Technol 17(6):541–557. https://doi.org/10.1016/j.trc.2008.07.002
34. Attanasio A, Bregman J, Ghiani G, Manni E (2007). Real-time fleet management at Ecourier Ltd. In: Zeimpekis V, Tarantilis CD, Giaglis GM, Minis I (eds) Dynamic fleet management, volume 38 of operations research/computer science interfaces, chapter 10, pp 219–238. Springer, New York
35. Godfrey G, Powell WB (2002) An adaptive dynamic programming algorithm for dynamic fleet management, I: single period travel times. Transp Sci 36(1):21–39
36. Powell WB, Topaloglu H (2005) Fleet management. In: Wallace S, Ziemba W (eds) Applications of stochastic programming, volume 5 of MPS-SIAM series on optimization, chapter 12. SIAM, pp 185–215
37. Simao H, Day J, George A, Gifford T, Nienow J, Powell WB (2009) An approximate dynamic programming algorithm for large-scale fleet management: a case application. Transp Sci 43(2):178–197
38. Du T, Wang FK, Lu P-Y (2007) A real-time vehicle-dispatching system for consolidating milk runs. Transp Res Part E Logist Transp Rev 43(5):565–577. https://doi.org/10.1016/j.tre.2006.03.001
39. Taniguchi E, Thompson R (2002) Modeling city logistics. Transp Res Rec J Transp Res Board 1790(1):45–51
40. Barcelo J, Grzybowska H, Pardo S (2007) Vehicle routing and scheduling models, simulation and city logistics. In: Zeimpekis V, Tarantilis CD, Giaglis GM, Minis I (eds) Dynamic fleet management. Operations research/computer science interfaces, vol 38. US, Springer, pp 163–195
41. Zeimpekis V, Minis I, Mamassis K, Giaglis GM (2007) Dynamic management of a delayed delivery vehicle in a city logistics environment. In: Zeimpekis V, Tarantilis CD, Giaglis GM, Minis I (eds) Dynamic fleet management, volume 38 of operations research/computer science interfaces series, chapter 9. Springer, New York, pp 197–217
42. Gendreau M, Guertin F, Potvin J-Y, Séguin R (2006) Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-up sand deliveries. Transp Res Part C Emerg Technol 14(3):157–174. https://doi.org/10.1016/j.trc.2006.03.002
43. Ghiani G, Manni E, Quaranta A, Triki C (2009) Anticipatory algorithms for same-day courier dispatching. Transp Res Part E Logist Transp Rev 45(1):96–106. https://doi.org/10.1016/j.tre.2008.08.003

44. Bieding T, Görtz S, Klose A (2009) On line routing per mobile phone a case on subsequent deliveries of newspapers. In: Nunen JA, Speranza MG, Bertazzi L (eds) Innovations in distribution logistics. Lecture notes in economics and mathematical systems, vol 619. Berlin, Springer, pp 29–51
45. Campbell A, Savelsbergh M (2005) Decision support for consumer direct grocery initiatives. Transp Sci 39(3):313–327
46. Ferrucci F, Bock S, Gendreau M (2011) Real-time distribution of perishable goods using past request information to anticipate future requests. Oper Res 34
47. Azi N, Gendreau M, Potvin JY (2011) A dynamic vehicle routing problem with multiple delivery routes. Ann Oper Res 13 (in press)
48. Balev S, Guinand F, Lesauvage G, Olivier D (2009) Dynamical handling of straddle carriers activities on a container terminal in uncertain environment—a swarm intelligence approach. In: Proceedings of the 2009 international conference on complex systems and applications (ICCSA 2009), Le Havre, France. University of Le Havre
49. Berbeglia G, Cordeau J-F, Laporte G (2010) Dynamic pickup and delivery problems. Eur J Oper Res 202(1):8–15. https://doi.org/10.1016/j.ejor.2009.04.024
50. Smolic-Rocak N, Bogdan S, Kovacic Z, Petrovic T (2010) Time windows based dynamic routing in multi-agv systems. IEEE Trans Autom Sci Eng 7(1):151–155. https://doi.org/10.1109/TASE.2009.2016350
51. Fiegl C, Pontow C (2009) Online scheduling of pick-up and delivery tasks in hospitals. J Biomed Inform 42(4):624–632. https://doi.org/10.1016/j.jbi.2009.02.003
52. Caramia M, Italiano G, Oriolo G, Pacifici A, Perugia A (2002) Routing a fleet of vehicles for dynamic combined pick-up and deliveries services. In: Proceedings of the symposium on operation research 2001, Duisburg, Germany, pp 3–5
53. Beaudry A, Laporte G, Melo T, Nickel S (2010) Dynamic transportation of patients in hospitals. OR Spectrum 32:77–107. https://doi.org/10.1007/s00291-008-0135-6
54. Kergosien Y, Lenté C, Piton D, Billaut J-C (2011) A tabu search heuristic for the dynamic transportation of patients between care units. Eur J Oper Res. https://doi.org/10.1016/j.ejor.2011.04.033
55. Romero M, Sheremetov L, Soriano A (2007) A genetic algorithm for the pickup and delivery problem: an application to the helicopter offshore transportation. In: Theoretical advances and applications of fuzzy logic and soft computing, volume 42 of advances in soft computing. Springer, Berlin, pp 435–444
56. Powell WB (2007) Approximate dynamic programming: solving the curses of dimensionality, volume 703 of Wiley series in probability and statistics. Wiley, Hoboken
57. Kilby P, Prosser P, Shaw P (1998) Dynamic VRPs: a study of scenarios. Technical Report APES-06-1998, University of Strathclyde, Glasgow, Scotland
58. Yang J, Jaillet P, Mahmassani H (2004) Real-time multivehicle truckload pickup and delivery problems. Transp Sci 38(2):135–148. https://doi.org/10.1287/trsc.1030.0068
59. Chen Z, Xu H (2006) Dynamic column generation for dynamic vehicle routing with time windows. Transp Sci 40(1):74–88
60. Montemanni R, Gambardella LM, Rizzoli AE, Donati AV (2005) Ant colony system for a dynamic vehicle routing problem. J Combin Optim 10(4):327–343. https://doi.org/10.1007/s10878-005-4922-6
61. Gambardella L, Rizzoli A, Oliverio F, Casagrande N, Donati A, Montemanni R, Lucibello E (2003) Ant colony optimization for vehicle routing in advanced logistics systems. In: Proceedings of the international workshop on modelling and applied simulation (MAS 2003), pp 3–9
62. Rizzoli A, Montemanni R, Lucibello E, Gambardella L (2007) Ant colony optimization for real-world vehicle routing problems. Swarm Intell 1:135–151
63. Taillard ED, Gambardella LM, Gendreau M, Potvin J-Y (2001) Adaptive memory programming: a unified view of metaheuristics. Eur J Oper Res 135(1):1–16. https://doi.org/10.1016/S03772217(00)00268-X
64. Ichoua S, Gendreau M, Potvin J-Y (2000) Diversion issues in real-time vehicle dispatching. Transp Sci 34(4):426–438. https://doi.org/10.1287/trsc.34.4.426.12325
65. Ichoua S, Gendreau M, Potvin J-Y (2003) Vehicle dispatching with time-dependent travel times. Eur J Oper Res 144(2):379–396. https://doi.org/10.1016/S0377-2217(02)00147-9
66. Romero M, Sheremetov L, Soriano A (2007) A genetic algorithm for the pickup and delivery problem: an application to the helicopter offshore transportation. In: Castillo O, Melin P, Ross OM, Sepúlveda Cruz R, Pedrycz W, Kacprzyk J (eds) Theoretical advances and applications of fuzzy logic and soft computing. Advances in soft computing, vol 42. Springer, Berlin
67. Al Chami Z, Manier H, Manier M-A, Fitouri C (2017) A hybrid genetic algorithm to solve a multi-objective Pickup and Delivery Problem. IFAC-Papers OnLine 50(1):14656–14661. https://doi.org/10.1016/j.ifacol.2017.08.1906
68. Powell WB, Sheffi Y, Nickerson KS, Butterbaugh K, Atherton S (1988) Maximizing profits for North American Van Lines' truckload division: a new framework for pricing and operation. Interfaces 18(1):21–41
69. Thomas BW, White CCI (2004) Anticipatory route selection. Transp Sci 38(4):473–487. https://doi.org/10.1287/trsc.1030.0071
70. Powell WB, Bouzaiene-Ayari B, Simao H (2007) Dynamic models for freight transportation. In: Barnhart C, Laporte G (eds) Transportation, volume 14 of handbooks in operations research and management science, chapter 5. North-Holland, pp 285–365
71. Yang S, Hamedi M, Haghani A (2005) Online dispatching and routing model for emergency vehicles with area coverage constraints. In: Network modeling 2005, number 1923 in transportation research record, pp 1–8
72. Haghani A, Yang S (2007) Real-time emergency response fleet deployment: concepts, systems, simulation and case studies. In: Zeimpekis V, Tarantilis CD, Giaglis GM, Minis I (eds) Dynamic fleet management. Operations research/computer science interfaces, vol 38. Springer, New York, pp 133–162
73. Solomon MM (1987) Algorithms for the vehicle-routing and scheduling problems with time window constraints. Oper Res 35(2):254–265
74. Flatberg T, Hasle G, Kloster O, Nilssen EJ, Riise A (2007) Dynamic and stochastic vehicle routing in practice. In: Zeimpekis V, Tarantilis CD, Giaglis GM, Minis I (eds) Dynamic fleet management, volume 38 of operations research/computer science interfaces, vol 38. US, Springer, pp 41–63
75. Pillac V, Guéret C, Medaglia AL (2012) An event-driven optimization framework for dynamic vehicle routing. Decis Support Syst. https://doi.org/10.1016/j.dss.2012.06.007
76. Hvattum LM, Lokketangen A, Laporte G (2006) Solving a dynamic and stochastic vehicle routing problem with a sample scenario hedging heuristic. Transp Sci 40(4):421–438. https://doi.org/10.1287/trsc.1060.0166
77. Mitrović-Minić S, Krishnamurti R, Laporte G (2004) Double-horizon based heuristics for the dynamic pickup and delivery problem with time windows. Transp Res Part B Methodol 38(8):669–685. https://doi.org/10.1016/j.trb.2003.09.001

78. Branke J, Middendorf M, Noeth G, Dessouky M (2005) Waiting strategies for dynamic vehicle routing. Transp Sci 39(3):298–312. https://doi.org/10.1287/trsc.1040.0095
79. Thomas BW (2007) Waiting strategies for anticipating service requests from known customer locations. Transp Sci 41(3):319–331. https://doi.org/10.1287/trsc.1060.0183
80. Ghiani G, Laporte G, Manni E, Musmanno R (2008) Waiting strategies for the dynamic and stochastic traveling salesman problem. Int J Oper Res 5(4):233–241
81. Bent R, Van Hentenryck P (2007) Waiting and relocation strategies in online stochastic vehicle routing. In: Veloso M (ed) Proceedings of the 20th international joint conference on artifical intelligence (IJCAI-07), pp 1816–1821
82. Branchini RM, Armentano VA, Lokketangen A (2009) Adaptive granular local search heuristic for a dynamic vehicle routing problem. Comput Oper Res 36(11):2955–2968. https://doi.org/10.1016/j.cor.2009.01.014
83. Bent R, Van Hentenryck P (2007) Waiting and relocation strategies in online stochastic vehicle routing. In: IJCAI international joint conference on artificial intelligence, pp 1816–1821
84. RamachandranPillai R, Arock M (2019) An adaptive spiking neural P system for solving vehicle routing problems. Arab J Sci Eng 1–17
85. Larsen A (2000) The dynamic vehicle routing problem. Kgs. Lyngby, Technical University of Denmark (DTU). IMM-PHD, No. 2000-73, Denmark
86. Haitao X, Pan P, Duan F (2018) Dynamic vehicle routing problems with enhanced ant colony optimization. Discrete Dyn Nat Soc. https://doi.org/10.1155/2018/1295485
87. Ionescu M, Păun G, Yokomori T (2006) Spiking neural P systems. Fundam Inf 71(2):279–308
88. Qi F, Liu M (2018) Optimization spiking neural P system for solving TSP. https://doi.org/10.1007/978-3-319-73447-7_71
89. Yang XS (2009) Firefly algorithms for multimodal optimization. In: Stochastic algorithms: foundations and applications. Springer, pp 169–178
90. Yang XS (2008) Nature-inspired metaheuristic algorithms. Luniver Press, London
91. Zeng X, Song T, Pan L, Zhang X (2011), Spiking Neural P systems for arithmetic operations. In: 2011 sixth international conference on bio-inspired computing: theories and applications, Penang, pp 296–301. https://doi.org/10.1109/bic-ta.2011.42
92. Paun G, Pérez-Jiménez M, Rozenberg G (2006) Spike trains in spiking neural P systems. Int J Found Comput Sci 17:975–1002. https://doi.org/10.1142/S0129054106004212
93. Rong H, Wu T, Pan L, Zhang G (2018) Spiking Neural P systems: theoretical results and applications: essays. Dedicated to Mario de Jesús Pérez-Jiménez on the Occasion of His 70th Birthday. https://doi.org/10.1007/978-3-030-00265-7_20
94. Metta VP, Kelemenová A (2015) Sorting using spiking neural P systems with anti-spikes and rules on synapses. In: Rozenberg G, Salomaa A, Sempere J, Zandron C (eds) Membrane computing. CMC 2015. Lecture Notes in Computer Science, vol 9504. Springer, Cham
95. Zein M, Adl A, Ella Hassanien A (2018), Spiking neural P grey wolf optimization system: Novel strategies for solving non-determinism problems, Expert systems with applications, volume 121, 2019, pp 204–220. https://doi.org/10.1016/j.eswa.2018.12.029
96. Xu H, Pu P, Duan F (2018) Dynamic vehicle routing problems with enhanced ant colony optimization. Discrete Dyn Nat Soc. https://doi.org/10.1155/2018/1295485
97. Khouadjia MR, Sarasola B, Alba E, Jourdan L, Talbi E-G (2012) A comparative study between dynamic adapted PSO and VNS for the vehicle routing problem with dynamic requests. Appl Soft Comput 12(4):1426–1439
98. Hanshar FT, Ombuki-Berman BM (2007) Dynamic vehicle routing using genetic algorithms. Appl Intell 27(1):89–99
99. Bell JE, McMullen PR (2004) Ant colony optimization techniques for the vehicle routing problem. Adv Eng Inf 18(1):41–48
100. Yu B, Yang ZZ, Yao BZ (2011) A hybrid algorithm for vehicle routing problem with time windows. Expert Syst Appl 38(1):435–441. https://doi.org/10.1016/j.eswa.2010.06.082
101. Tan CK, Chew YH, Lee LH (2006) A hybrid multiobjective evolutionary algorithm for solving vehicle routing problem with time windows. Comput Optim Appl 34:115–151. https://doi.org/10.1007/s10589005-3070-3
102. Dong W, Zhou K, Qi H, He C, Zhang J (2018) A tissue P system based evolutionary algorithm for multi-objective VRPTW. Swarm Evolut Comput 39:310–322. https://doi.org/10.1016/j.swevo.2017.11.001
103. Zhao H (2007) A multi-objective genetic programming approach to developing Pareto optimal decision trees. Decis Support Syst 43(3):809826. https://doi.org/10.1016/j.dss.2006.12.011
104. Haynes W (2013) Holm's method. In: Dubitzky W, Wolkenhauer O, Cho KH, Yokota H (eds) Encyclopedia of systems biology. Springer, New York
105. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evolut Comput 1(1):3–18. https://doi.org/10.1016/j.swevo.2011.02.002
106. Chen SY, Feng Z, Yi X (2017) A general introduction to adjustment for multiple comparisons. J Thor Disease 9(6):1725–1729. https://doi.org/10.21037/jtd.2017.05.34

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.