Journal of Computers Vol. 29 No. 1, 2018, pp. 112-131
doi:10.3966/199115992018012901010

A Membrane-based Evolution Algorithm with Time Classifier for VRPTW

Yingying Duan[1], Kang Zhou[1*], Huaqing Qi[2], Hui Zhang[3], Xin Tong[1]

[1] School of Math and Computer, Wuhan Polytechnic University, 430023 Wuhan, China
dybngduan@163.com, zhoukang65@whpu.edu.cn, tx895146920@sina.com

[2] Department of Economics and Management, Wuhan Polytechnic University, 430023Wuhan, China
qihuaqing@sohu.com

[3] School of Information and Intelligence Engineering, Ningbo City College of Vocational Technology, 315000 Ningbo, China
1094756649@qq.com

Received 1 August 2017; Revised 20 October 2017; Accepted 20 November 2017

Abstract. Vehicle routing problem with time window is of profound theoretical research significance and broad practical application value.we propose a membrane algorithm with genetic mechanism to improve the convergence speed or population diversity, because traditional heuristics still have shortcomings in these two problems. In this algorithm, we introduce membrane techniques to increase the diversity of population. We put forward time classifier to further accelerate the evolving speed of each membrane. We propose a new crossover operator in order to further improve the successful probability of crossover operator; Beside,we can also designed an improved roulette mechanism so as to modify the quality of solutions. For membrane algorithm, its most prominent advantage is that the distribution and the parallelism can improve both he ability of local search and global search and the efficiency of this algorithm. The experimental results shows that membrane algorithm with genetic mechanism is competitive with other heuristics.

Keywords: improved crossover operator, membrane computing, time classifier, vehicle routing problem with time window

1 Introduction

Vehicle routing problem (VRP) is an NP-hard combinatorial optimization problem [1]. Vehicle routing problem with time windows (VRPTW) is an expand of VRP, which adds time windows to each client and specifies that each vehicle must be visited within this window. VRPTW has important theoretical and practical significance in logistic research area such as rail distribution system, the mail and newspaper delivery system and so on. So far, the variant of VRPTW has been widely studied. For example, papers [2-3] studied VRPTW with dynamic constraint and time-dependent, paper [4] studied the hard time window VRPTW, papers [5-6] studied separately the soft time window problems. Because heuristic algorithms has been proved that it could reduce computational complexity in solving optimization problems by doing a lot of research in this aspect by scholars, many fields used heuristic algorithms to solve some practical problems, for instance, papers [7-8] studied branch-and-price algorithm and artificial bees algorithm are proposed to optimize VRPTW with different constraints, papers [9-11] proposed some improved ant colony algorithms to solve this problem. However, further studying shows that genetic algorithm is still of high computational cost in efficiency or precision, especially for the large-scale problems. Therefore we introduce membrane thought to solve the problems encountered.

* Corresponding Author

Membrane computing is a new and vigorous branch of natural computing. The computing models in membrane computing are called membrane systems or P systems, which are parallel and distributed systems. The theoretical study in membrane computing mainly concentrates in abstracting all kinds of variants from the structure and functioning of cells and investigating the computational power of these variants. According to the membrane structure, there are two main families of P systems: cell-like P systems which have a hierarchy arrangement of membranes as in a cell [12]; and tissue-like P systems or neural-like P systems (spiking neural P systems, SN P systems) as in a tissue inspired by cell intercommunication in tissue [13] or in a neural net inspired by some way neurons communicate with each other by means of electrical impulses of identical shape [14]. Many variants of P systems are proposed by introducing other biological features into the system, such as P systems with active membranes [15], P systems with string objects [16], axon P systems [17], SN P systems with structural plasticity [18], SN P systems with request rules [19], SN P systems with anti-spikes [20-21], SN P systems with weights [22-23], SN P systems with rules on synapses [24-25], SN P systems with white whole neurons [26], cell-like SN P systems [27]. Many strategies of using rules are considered in membrane computing, including minimal parallel [28], time-freeness [29], asynchronous [30], flat maximal parallel [31], sequential [32]. Most of these variants of P systems working in different modes are proved to be Turing universal [33-35]. P systems are able to solve computationally hard problems in a feasible time [36]. Recently, P systems have been sused to solve NP-complete problems in a time-free manner in the sense that the correctness of the solution does not depend on the precise timing of the involved rules [37], such as to solve SAT problem [38], common algorithmic problem [47] and 3-coloring problem [39]. P systems have been applied in many fields [40-41], such as biology, linguistics, theoretical and applied computer science [42-45]. The recent applications in engineering can be found in [46-48]. For the most up-to-date news and results, one can consult the P systems web page http://ppage. psystems.eu.

Based on the advantages of P system above mentioned, some scholars proposed the combination membrane computing and heuristics to solve some practical problems. Papers [49-50] applied membrane algorithms with different heuristics such as tabu algorithm, particle swarm algorithm and so on to solve traveling salesman problems. Paper [51] used membrane algorithm with genetic mechanism to solve vehicle routing problem. Paper [52-54] optimizes solid waste transportation problem by using membrane algorithm with a three-levels of hierarchical cell-like structure. Papers [55-57] analyzed the diversity and convergence of membrane algorithms and its implementation on GPU, which produces the better solutions by comparing with the heuristic by dividing a large problem into several sub-problems to process. However, very few literature emphasize the effect on the experimental results of membrane framework design. In this work, we propose a membrane algorithm owned a certain structure with genetic mechanism. In this algorithm, we propose some strategies to enhance the performance of the algorithm such as for efficiency we introduce time classifier and for precision we introduce an uncertain-segment crossover and the segment-node insertion operation. These strategies can effectively improve the efficiency or the quality of solutions on the basis of membranes thus improving the whole performance of membrane algorithm. The paper is organized as follows: Section 2 describes the definition and mathematical model; Section 3 is devoted to the object and evolution rules; Section 4 addresses the experimental design and presents computational results based on standard data sets. Finally, conclusions are presented in Section 6.

2 Mathematical Formulation for VRPTW

The vehicle routing problem with time windows can be defined as a problem of finding a minimum cost route for a number of homogeneous vehicles stationed at a depot, which have the task of delivering goods to a number of customers under the premise of satisfying the following constraints such that:

- The route of each vehicle must start and end at a depot.
- Each customer must be serviced by only one vehicle.
- The demand of all customers must be delivered.
- The route of no vehicle should contain a customer demand sum greater than the vehicle capacity.
- Each vehicle must arrive at the customer delivery sites within the time horizons associated with them, it may arrive earlier, but then must wait.

Mathematically, it can be described as: the problem is given by a set of customers $V = \{v_1, v_2, ..., v_n\}$, which each customer is associated with a physical location specified by x and y coordinates, where the distance between customers is given by the Euclidean distance. Let v_0 denote the location of the depot, $V' = V \cup \{v_0\}$ becomes the set of all locations considered in the problem, $N = \{1, 2, ..., n\}$ becomes a natural number set that represents the position of v_i. Every pair of locations (i, j), where $i, j \in N$ and $i \neq j$, is associated with a cost of travelling d_{ij} and a travel time t_{ij}, where the travel time between two customers is assumed to be equal to the distance. For each customer v_i, there is a demand q_i, a service time s_i, and a service time window $[e_i, l_i]$ where e_i describes the earliest time to start servicing and l_i describes the latest time to start servicing. The demand q_i of customer i is to be serviced by exactly one vehicle within its time window. In addition, e_0 describes the earliest time to depart depot and l_0 describes the latest time to arrive depot. The objective is to determine a feasible route schedule which primarily minimizes the total travel distance.

$$\text{Min } Z = \sum_{k=1}^{K} \sum_{i=0}^{n} \sum_{j=0}^{n} d_{ij} x_{ijk} \tag{1}$$

Subject to:

$$\sum_{j=1}^{n} x_{0jk} \leq 1, \quad k = 1, 2, ..., K \tag{2}$$

$$\sum_{k=1}^{K} \sum_{i=0}^{n} x_{ijk} = 1, \quad j = 1, 2, ..., n \tag{3}$$

$$\sum_{j=1}^{n} x_{ijk} - \sum_{j=0}^{n} x_{jik} = 0, \quad i = 1, 2, ..., n; k = 1, 2, ..., K \tag{4}$$

$$\sum_{i=1}^{n} q_i \left(\sum_{i=0}^{n} x_{ijk} \right) \leq Q, \quad k = 1, 2, ..., K \tag{5}$$

$$\sum_{k=1}^{K} \sum_{i,j=S} x_{ijk} \leq |S| - 1, \quad S \subseteq N, 2 \leq |S| \leq n - 2 \tag{6}$$

$$w_i = \max \{0, e_i - a_i\}, \quad i = 1, 2, ..., n \tag{7}$$

$$e_i \sum_{j=1}^{n} x_{ijk} \leq a_i + w_i \leq l_i \sum_{j=1}^{n} x_{ijk}, \quad k = 1, 2, ..., K \tag{8}$$

$$x_{ijk} \in \{0, 1\}, \quad i, j = 1, 2, ..., n; k = 1, 2, ..., K \tag{9}$$

$$\sum_{j=1}^{n} x_{0jk} \leq 1, \quad k = 1, 2, ..., K \tag{10}$$

where the decision variable a_i is denoted as the time of arriving at customer v_i serviced by vehicle k ($k \in \{1, 2, ..., K\}$); the decision variable w_i is the time of waiting at customer v_i; the decision variable x_{ijk} is whether the route from v_i to v_j is serviced by vehicle k: if $x_{ijk} = 1$, then e ij is visited by vehicle k; else, e_{ij} is visited by other vehicles; K is the number of vehicles.

The goal of the objective function (1) is to minimize the total cost. Formula (2) ensures that there is at most a vehicle from the depot v_0 to a client v_j. Formula (3) makes certain that each customer is visited by exactly a vehicle. Formula (4) describes that a vehicle arriving to a customer is the same as that leaving it. Formula (5) ensures that the sum of customer demands assigned to a specific vehicle does not exceed its maximum capacity. Formula (6) ensures that each routes cannot exist ring. Formula (7) computes the

wait time of this vehicle in customer v_i. Formula (8) describes that the arrival time of the latter is less than one of the former in the same route (M is a larger scalar). Formula (9) describes that the service time of a client must be in its windows. Formula (10) guarantees the non-negativity of variables x_{ijk} and define the Formulation as a binary integer linear programming model, where if i is equal to j, then x_{ijk}=0

3 Hybrid Evolution Mechanism Driven By Membrane Computing

In this section, we propose a membrane algorithm with genetic evolution mechanism (MGA), which is the integrant of tissue P system and genetic algorithm. MGA introduces different evolution mechanisms into membranes such as uniform order crossover or partially mapped crossover. Each individual in genetic algorithm is mapped as an object. The evolution mechanism is regarded as the rules of each membrane. There are some communications between membranes. In order to better introduce how genetic algorithm is combined with P system and how to solve vehicle routing problem with time window, the structure, objects and rules of membrane algorithm are introduced as follows.

3.1 Membrane Optimization Framework

Problem decomposition schemes have been in place for quite some time. The most popular and newest method using this scheme is called membrane system, introduced by the professor Păun [15]. The system works in the following manner.

- Break the problem into smaller sub-problems.
- Solve the sub-problem.
- Construct the global problem from the sub-problem.
- Repeat the above 3 steps.

The same strategy was used by Nishida in tabu search for the TSP in 2005 [58]. He partitioned a whole membrane space into independent sub-space, optimized the objects, then joined them together to construct the global solution, only to partition it again, into different area of sub-space. However, different operations produces different membrane construct methods. The membrane structure was extended in [45] with a cell-like mode which ensured different in each iteration. We have further generalized this structure into the localized optimization framework. For MGA, we propose a decomposition strategy based on different operators, which independently search to every neighborhood space. A membrane system of degree 7 is a construct.

$$\prod = (\mathrm{O}, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \sigma_6, \mathrm{syn}, i_0) \tag{11}$$

where,

(1) $O = \{X_0, X_1, ..., X_{S-1}\}$, X_i (i = 0, 1, ..., S−1) is a string (object), S is the number of objects;

(2) syn={(1, 3), (1, 4), (1, 5), (2, 3), (2, 4), (2, 5), (3, 2), (4, 2), (5, 2), (2, 6), (6, 2)};

(3) i_0 =0 is an output membrane labeled as 0;

(4) σ_1, ..., σ_6, are cells, of the form

$$\sigma_i = (Q_i, s_i, 0, \omega_i, 0, R_i),\ 1 \le i \le 6, \tag{12}$$

where:

(a) $Q_i = (s_{i,0}, s_{i,1}, ..., s_{i,k}, t\_max)$, $s_{i,t}$ is the tth iteration of the ith cell, t_max is a max iteration.

(b) $\omega^t_{k,li} = \{X^t_{k,1}, X^t_{k,2}, ..., X^t_{k,l\,i}\}$ is an object multi-set of the kth membrane;

(c) R_i is a rule set containing evolution rules and communication rules;

According to the framework of Fig. 1, this system executes as follows: firstly, generate S objects using the rule of cell 2; secondly, l_1, l_2 and l_3 objects are separately selected with replacement sampling from cell 2 and then sent to cell 3-5 using communication rules; l_4 ($l_4 = S-l_1-l_2-l_3$) objects selected with replacement sampling using roulette method are sent to cell 6 according to the same rules; finally, objects of membranes begin to evolve before synchronously resenting to cell 2 after evolving, complete a state transition, continue until reaching termination criterion, end.

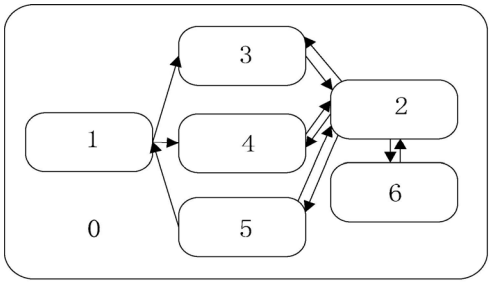

Fig. 1. The framework of MGA

The above system introduces the framework of membrane system, objects and rules contained the evolutionary rules and the transportation, where the framework makes the objects of each cell evolving correctly in a closed and independent environment. In this system, membranes evolved expand the search space of population due to the different operations of four membranes and the design of cell 1 deduces the running time because of optimizing the computational complexity of the algorithm, compared with heuristics in optimizing VRPTW. The system is beneficial to further improve the capacity of solving problems, which shows the effectiveness of this framework.

3.2 Formulation of Object Encoding for the VRPTW Problem

An object is generated by choosing a random ordering of nodes, forming a list of length N where each node appears once only (the depot is not included). a chromosome represents an order of customers without route splitters, for example, X_i=$\{x_{i1}, x_{i2}, ..., x_{iN}\}$(i∈[0, S−1). Due to the limitations of carrying capacity for each vehicle, it is usually impossible to complete all delivers using one truck, and therefore several journeys, or routes are necessary. As this representation does not include the information on where the list is broken by returns to the depot, or terminal points, an object for VRPTW is represented as: X_i = {0, x_{i1}, ..., x_{ik}, 0, ..., x_{iN}, 0} where it is a route between depots visited by only one vehicle, and the current state of each membrane is expressed as: $\omega_i = (X_{0i}, X_{1i}, ..., X_{(n-1)i})$ (n∈[1, S]), where x_{ij}∈[1, N], X_{ki} (k∈[0, n−1]) is a code of kth object in cell i.This encoding method can not only directly express a solution of VRPTW, but also intuitively select current client number x_{ij}, thus avoiding the operation of decoding and further improving the efficiency.

3.3 Formation and Optimization of Internal Membranes

In order to ensure the feasibility of the objects of each membrane, some rules are designed to control the evolution of objects. In MGA, the rules are divided two classes: (1) Evolution rules: cell 1 executes a classed operation; cell 2 generates an initial population, cell 3 and cell 4 execute different crossover operations, cell 5 executes mutation operation and cell 6 executes an optimum-recorded operation; (2) Transportation rules: the communication of membranes. The detailed introduction is as 3.3.1-3.3.6.

3.3.1 Rules in Cell 1: Time Classifier

In order for further enhancing the computing efficiency of membranes, a classed set T ($T = T + T_k$, k = 0, 1, ..., 23) is computed using time classifier introduced by cell 1. This mechanism is described as: time difference for the open and close time of depot is divided into twenty-four parts, where each partis recorded using a set T_k and its time interval is [t_k, t_{k+1}), all clients are classedand stored in these sets T_0, T_1, ..., T_{23} according to their time window, where if window [e_i, l_i] of a client vi satisfies the conditions: tk < e_i and li < tk+l, then client vi is classed respectively in these sets from T_k to T_k+l. In order to further research the nature of time classifier by combining with the characteristic for VRPTW, six kinds of lemmas for this mechanism are introduced as follows:

Lemma 1. In time classifier, for time window [e_i, l_i]: if $l_i > a_{i-1}$, then $u(x + v_i, v_i, y) \rightarrow v(x, v_i, y + v_i)$; else, v_i is directly deleted.

Lemma 2. *For crossover operator, execute the rules* $u_1 (x+s_1, s_1, y) \rightarrow u'_1(x_1+s_2, s_2, y_1)$ *and* $u_2 (x+s_2, s_2, y) \rightarrow u'_2(x_2+s_1, s_1, y_2)$*, that all segments in Time Classifier satisfy* s_1∈[e_{x2}, l_{y2}] *and* s_2∈[$e\ x_1$, ly_1] *can execute crossover operator.*

Lemma 3. *Initialize routing structure car_client, for arbitrary window* $[e_i, l_i]$, $[e_{i+1}, l_{i+1}]$ *and* $e_{i+1} > l_i$*: if select tail insertion operation, then these clients of Time Classifier that latest service* $l_k (k \in [0, i])$ *is less than e* $i+1$ *are directly deleted.*

Lemma 4. *For* v_i *of set* T_{N-1} (N *is the number of set*) *in Time Classifier, if* $(l_i - e_i) \cap (l_{i+1} - e_{i+1}) = \phi$ *or* $(l_i - e_i) \cap (l_{i+1} - l_{i+1}) \leq \varepsilon$*, then* $u(x+v_i, v_i, y) \rightarrow u'(x, v_i, y+v_i)$*, promptly, the probability that* v_i *is inserted to the tail of each vehicle is larger than the clients of other sets.*

Lemma 5. *For an inserted position, that a node v i satisfied constraint conditions of inserting this position must be between set* T_{k-1} *and* T_{k+1} *can ensure the feasibility of route after inserting* v_i*, promptly, and these nodes are directly excluded beyond time window.*

Lemma 1 shows that the latest service time of a node must be larger than depart time of previous node in the same path. Lemma 2 and Lemma 3 express that two objects can exchange as long as the windows are met. Lemma 4 illustrates that the probability is very smaller which nodes of T i are inserted to the back of these nodes belonged to T_k ($k > i$) in time classifier. Lemma 5 verifies that the probability of inserting to route tail for this kind of nodes owned a greater latest service time is larger. Lemma 6 explains that v_i to be inserted to position p_i must exist between T_k and T_{k+1}.

From this we can obtain that the algorithm can quickly produce a new object by using the rules of Lemma 1~5, that is, it can decrease computational complexity by directly excluding these clients not stored in time range of this position thus selecting some feasible clients satisfied window constraints to insert or swap, which proves that the design of time classifier is successful for further improving the computing efficiency of MGA, compared with the evolved operations that a client is selected and inserted to route R_t ($t \in [0, |R_t|]$ where $|R_t|$ expresses the number of all clients of the tth route), and then adjust whether all clients of R_t satisfied time window: if meet, continue; else, deleted and select.

3.3.2 Rules in Cell 2: Initialization Objects

Because MGA is a multi-population algorithm, S objects need to be generated by applying the corresponding technologies. For each object, the rule of coding decides that we have adopted the split procedure, which works through the customer sequence embedded in the object. It evaluates all feasible route sets that can be drawn from that customer sequence, and extracts the route set out of it, which has the minimal cost. To fulfill this task, we use a tail insertion to produce a feasible solution. In order for making the algorithm producing multiple objects in the closed and independent domain, cell 2 is used as a computing device for obtaining objects and stores new object set after each iteration. The process of generating objects is introduced as follows.

Tail-insertion Method. To fully apply the evolution rules for MGA, initial phrase, a randomly chosen customer is taken out from time classifier and reinserted to the end of a randomly chosen route under satisfying load and time windows of route.

The process of generating S objects. In initial phrase, set $T(T = \{T_0, T_1, ..., T_{23}\})$ is obtained by using time classifier, set N_d records all the information of each customer contained arrive time, service time, and so on. Initial set G ($G = G + \{X_i\}$)($i \in [0, S-1]$) is an empty set, set F is used to store the customers of set T in order to avoid the repeated insert of customers in the phrase. The process is described as the following steps.

Input:

- Time classifier: T
- Node: N_d
- C_list: $G = G$

Output: An object set $G = \{X_0, X_2, ..., X_{S-1}\}$

Step 1) Set $N_d = \phi$ and $F = \phi$; let the number of objects be S, $s = 0$.

Step 2) Compute a classed set T ($T = T \cup T_i$, $i = 0, 1, ..., 23$) using node classifier.

Step 3) Insertion the first n customers of set F to different routes.

Step 3.1) Compute the number of vehicles according to the requirement of full-load as formula:

$$n = \begin{cases} Q/q, & if Q/q \ is \ an \ integer \\ \lceil Q/q \rceil + 1, & else \end{cases}$$

Step 3.2) n clients are randomly inserted to the first position of n routes and then delete these clients from set F.

Step 4) Randomly insert client v_i into a chosen route using a tail insertion:

Step 4.1) v_i is selected from set F and then execute $F = F - \{v_i\}$.

Step 4.2) Randomly select a route R_t ($t < n$).

Step 4.3) If v_i can be inserted to the end of R_k: then insert and update R_k; else, turn Step 4.2; If any route cannot be inserted, turn Step 4.4.

Step 4.4) Reopen a route R_t (t=n+1), and insert v_i to R_t, continue until set F becomes an empty set, obtain an object X_s and then execute $G = G+\{X_s\}$, s++.

Step 5) If reaching the stopping criteria, then stop and output G. Otherwise, go to Step 3.

In initialization, we use euclidean distance to compute arrival time of each customer. For the first n nodes, they are randomly inserted to n different routes; Set $K = \{v_{j1}, v_{j2}, ..., v_{jk}\}(v_{jk} \in F, k \in [1, n])$, for other nodes v_{j1} ($1 \in [n+1, N]$), randomly select a route R_t ($t \in [0, n-1]$), if $a_{j1} < l_{jk}$ ($v_{jk} \in K$) and $a_{jk} \in [e_{jk}, l_{jk}]$ under satisfying max-load, then v_{jk} is inserted to Rt, continue until N clients are inserted, end; obtain an object Xs. According to the method above mentioned, S objects are generated. It can be seen that a tail insertion can both avoid computing the windows behind current client and improve the crossover-succeeded (mutation) rate, which ensures the effectiveness of this algorithm.

3.3.3 Rules in Cell 3: Crossover Operator

Because traditional crossover operator is executed by swapping two ordered segments of different individuals, this operator can produce a plenty of unfeasible solutions due to the existence of time span in solving VRPTW.According to the characteristic of VRPTW, we design an improved crossover operator that is executed by calling an anterior operator or a tail operator so as to enhance the successful probability of crossover operator. In the following section, we firstly introduce the definition of three operators, secondly, we introduce the pseudo of three sub-algorithms, finally, according to the evolutionary process of the crossover, we design evolution rules of this cell.

Definition for an improved crossover operator. Because we need to apply two techniques in defining crossover operator, we firstly introduce the definitions of an anterior operator and a tail insertion before introducing crossover. The detailed descriptions are listed as follows.

Definition 1 *For an object* $X = (x_1, x_2, ..., x_n)$ *and an ordered segment* $s = <v_i, v_{i+1}>$, *if* $x_k = v_i$ *and* $x_j = v_{i+1}$ ($k < j$), *then* (*a*) *we called the operation* $Fn(X, s) = (x_1, x_2, ..., x_{k-1}, x_{k+1}, ..., x_k, x_j, ..., x_n)$ *as Anterior Insertion Operator; we called the operation* $Tn(X,s) = (x_1, x_2, ..., xk, x_j, ..., x_{j-1}, x_{j+1}, ..., x_n)$ *as Tail Insertion Operator.* (*b*) *If an object Fn* (*X, s*) *or Tn* (*X, s*) *satisfies the constraints of vehicle surplus and time window, then it is a feasible object.*

Definition 2 *For two objects* X_i *and* X_j, *two routes* R_1 *and* R_2 *are randomly selected from objects* X_i *and* X_j, *and two ordered segments* $s_1 = <v_k, v_{k+1}>$ *and* $s_2 = <v_t, v_{t+1}>$ *are arbitrarily selected from routes* R_1 *and* R_2. *For the objects of crossover, we need to adjust:*

(a) If an object $Fn(X_i, s_2)$ *or* $Tn(X_i, s_2)$ *is a feasible object, then it is the child of* X_i *after exchanging the segments of* X_i *and* X_i; *else,* X_i *is the child of* X_i *generated after executing the crossover operator.*

(b) If an object $Fn(X_j, s_1)$ *or* $Tn(X_j, s_1)$ *is a feasible object, then it is the child of Xj after exchanging the segments of* X_i *and* X_j; *else,* X_j *is the child of* X_j *generated after executing the crossover operator.*

The above definition describes a new crossover operator, this operator can be not only easy to be executed, but also inherit the excellent fragments of the previous individuals. The experimental results at the later stage can be used for verifying that the computational results of this operator can be accepted.

The pseudo for the improved crossover. In order to clearly describes the computational process of crossover factor,we give the following pseudo: where $A_{LGORITHM}$ 1 states the process of anterior insertion, $A_{LGORITHM}$ 2 lists the process of tail insertion, $A_{LGORITHM}$ 3 shows the total process of the crossover. In $A_{LGORITHM}$ 1 and $A_{LGORITHM}$ 2, *car_client*[*i*].*route*[*j*]. *number* is the number of a node in the *j*th route of the *i*th objects, a_{vt} is the arrival time of a node v_t, a route R_1 (R_2) is selected from an object X_1 (X_2).

ALGORITHM 1 An anterior insertion

Input: X_1, X_2 (two objects), R_1, R_2 (two routes), s_1, s_2 (two segments)

```
t = rand()/R2
for i = t ← t+1
s2 = s2 + {vi}
for i = 0 ← |X1|
for j = 0 ← |R1|
if car_client[i].route[j].number == vk+1
vt+1 = car_client[i].route[j].number
if (vt satisfies the condition of inserting the former of vt+1)
avt = car_client[i].route[j-1].arrive+C-Client(car_client[i].
   route[j-1].number, vt)
if(avt < lvt)
R1 = R1 + {vt}
```

Output: A new object X'_1

ALGORITHM 2 Tail insertion

Input: X_1, X_2 (two objects), R_1, R_2 (two routes), s_1, s_2 (two segments)

```
t = rand()/R2
for i = t ← t + 1
s2 = s2+{vi}
for i = 0 ← |X1|
for j = 0 ← |R1|
if (car_client[i].route[j].number == vt)
vt= car_client[i].route[j].number
if(vt+1 satisfies the condition of inserting the back of vt)
avt+1 = avt + C_Client(vt,vt+1)
if(avt+1 < lvt+1)
R1 = R1 + {vt+1}
```

Output: A new object X'_1

ALGORITHM 1 and ALGORITHM 2 introduce in detail that the process of the anterior or tail insertion. The difference between two insertion methods is the insertion location such that ALGORITHM 1 is used for inserting a node to the front of another node satisfying vehicle surplus and time windows as 6~9, and ALGORITHM 2 is used for inserting a node to the back of another node satisfying above constraints as 6~9. According to analysis, this method can not only in crease the probability of successful crossover, but also reduce its computational efficiency because it forms a fragment of another chromosome by inserting a single point using ALGORITHM 1 or ALGORITHM 2. ALGORITHM 3 describes the total process of crossover operation. In this algorithm, the first step is to randomly two fragments s_1 ($s_1 \subseteq X_1\}$) and s_2 ($s_2 \subseteq X_2\}$); the second one is to make the following operators: for the segment s_1, the locations of nodes v_k and v_{k+1} in a route X_2 are searched, if a selected node v_k can insert the front of v_{k+1} by calling ALGORITHM 1, then insert and form a segment s_1 in X_2; else, if v_{k+1} can be inserted the back of v_k by calling ALGORITHM 2, then insert and generate a segments 1 in X_2. The theory analysis states that this method can only need to adjust the inserted feasibility of a node, however traditional fragment trans-plantations can need to adjust the feasibility of nodes and the span of time windows, therefore this operator is effective in generating feasible solutions.

The proposition for the improved crossover. According to the definition of an improved crossover, the paper analyses the following proposition.

Proposition 1: *For crossover operator, if satisfy the rules* $u_1(x_1 + s_1, s_1, y_1) \rightarrow u'_1(x_1+s_2, s_2, y_1)$ *and* $u_2(x_2+s_2, s_2, y_2) \rightarrow u'_2(x_2+s_1, s_1, y_2)$*, then all segments satisfying* $s_1 \in [e_{x2}, l_{y2}]$ *and* $s_2 \in [e_{x1}, l_{y1}]$ *in Time Classifier can execute crossover operation.*

Proof: Let $R_1 = x_1 s_1 y_1$ and $R_2 = x_2 s_2 y_2$ where $R_1 \subseteq X_i$ ($i \in [0, S-1]$) and $R_2 \subseteq X_j$ ($j \in [0, S-1]$ & $i! = j$), $s_1 = \{v_m, ..., v_{m+l}\}$ and $s_2 = \{v_n, ..., v_{n+k}\}$ ($m = n$ or $m! = n$). Under the premise of max-load of each vehicle, for rule u_1 $(x_1+s_1, s_1, y_1) \rightarrow u'_1$ (x_1+s_2, s_2, y_1), if $l_m > a_{n-1}$ or $e_{m+l-1} < d_{n+k+1}$, then the above rule can be

executed and s_1 can be inserted to route R_2 thus ensuring that arrive time of all clients in current route R_1 is less than the latest service time; else, s_1 can not be inserted to R_2 because it can not satisfy time windows of s_2 in R_2; for rule u_2 (x_2+s_2, s_2, y_2) → u'_2 (x_2 +s_1, s_1, y_2), if $l_n > d_{m-1}$ or $l_{n+k-1} < d_{m+l+1}$, then this rule can execute and s_2 can be inserted to route R_1; else, s_2 can not be inserted to R_1, select a new segment and continue. In summary, the premise of the exchange of s_1 and s_2 must satisfy time windows of the exchanged route, if arrival time a_m of any node v_m of the inserted route is larger than l_m, then the rules can not be used. □

ALGORITHM 3 Crossover Operation: Obtaining two objects

Input: X_1, X_2 (two objects), R_1, R_2 (two routes), s_1, s_2 (two segments)

```
k = rand()/R1
t = rand()/R2
for i = k ← k + 1
s1 = s1 + {vi}
for i = t ← t + 1
s2 = s2 + {vi}
for i = 0 ← |X1|
if (vt can be inserted into the former of vt+1)
X'1 = Anterior_insertion(X1, s2);
else X'1 = Tail_insertion(X1, s1);
for i = 0 ← |X2|
if (vk can be inserted into the former of vk+1)
X'2 = Anterior_insertion(X2, s1);
else X'2 = Tail_insertion(X2, s1);
```

Output: A new object X'_1, X'_2

Rules for crossover mechanism. According to the previous introduction of the operator, we design the following rule: where select two objects X_i and X_j ($i, j \in [0, S-1]$) where $X_i = x_1 s_1 y_1$, $X_j = x_2 s_2 y_2$, and execute the following rules and obtain new objects X'_i and X'_j.

$$r_3 \equiv \{(x_1 s_1 y_1)_{\neg s2} \rightarrow (x_1 s_2 y_1)\} \tag{12}$$

$$r_4 \equiv \{(x_2 s_2 y_2)_{\neg s1} \rightarrow (x_2 s_1 y_2)\} \tag{13}$$

where $X_i = \{x_{i1}, x_{i2}, x_{i3}, ..., x_{i(m-1)}, x_{im}, x_{i(m+1)}, x_{i(m+2)}, ..., x_{i(m+k)}, ..., x_{iN}\}(s_1 = \{x_{im}, x_{i(m+1)}\})$, $X_j = \{x_{j1}, x_{j2}, x_{j3}, ..., x_{jn}, x_{j(n+1)}, ..., x_{j(n+p-1)}, x_{j(n+p)}, x_{j(n+p+1)}, ..., x_{j(n+p+t)}, x_{j(n+p+t+1)}, ..., x_{2N}\}(s_2 = \{x_{j(n+p)}, x_{j(n+p+1)}\})$, $s_{1(2)}$ is inserted to an object X_k ($k \in \{i, j\}$) by the form of the whole segment.

For formula (14), the example of crossover is as:

$$r_3 : \{0\ 9\ 2\ 3\ 0\ 8\ 4\ 1\ 5\ 0\ 7\ 6\ 0\}\ (8,4) \rightarrow \{0\ 9\ 6\ 2\ 3\ 0\ 8\ 4\ 1\ 5\ 0\ 7\ 0\}\ (9,6) \tag{14}$$

For formula (15), the example of crossover is as:

$$r_4 : \{0\ 4\ 1\ 0\ 2\ 9\ 6\ 0\ 7\ 8\ 0\ 3\ 5\ 0\}\ (9,6) \rightarrow \{0\ 1\ 0\ 2\ 9\ 6\ 0\ 7\ 8\ 4\ 0\ 3\ 5\ 0\}\ (8,4) \tag{15}$$

Formulas (14) and (15) show that new objects X'_i and X'_j ($i \in [0, S-1]$) are produced as ALGOGITHM 3: a segment s_1 (such as {8, 4}) is randomly selected from route R_k ($R_k \subseteq X_i$ & $k \in \{1, 2\}$), a crossover segment s_2 ({9, 6}) is obtained as: (1) initially compute these vehicles R satisfied max-load such as formula (5) where $R = R + R_{kl}$ ($R_{kl} \subseteq X_j$), $l \in [0, |K|]$ and $k_l \leq k$); (2) In time classifier, search the position of a node of $x_{i(n-1)}$ ($v_{n-1} \in T$)(8) and $x_{l(j+2)}$ ($v_{j+2} \in T$)(4) and $s^0{}_2$ (such as {8,4,1,5}) belonged to a chosen route R_k obtained between the classed sets T_1 and T_2; (3) $s^1{}_2$ ($s^1{}_2 \subseteq s^0{}_2$) is computed by adjusting whether satisfies the minimum delay time window such as formula (8) and (9) of R_{kl}: if exist, then $s_2 = s^1{}_2$ is a crossover segment of s_1; else, go to (1), select; After two crossover segments, s_1 and s_2 are directly exchanged and obtain new objects X'_i (0 → 9 → 6 → 2 → 3 → 0, 0 → 8 → 4 → 1 → 5 → 0, 0 → 7 → 0)($|K| = 3$) and X'_j (0 → 1 → 0, 0 → 2 → 9 → 6 → 0, 0 → 7 → 8 → 4 → 0 → 3 → 5 →0)($|K| = 4$).

In summary, this method can pass some excellent segments of parents into the next generation in order to make individuals evolve toward an optimal direction thus improving the precision for solutions on the basis of multiple membranes.

ALGORITHM 4 Swap_po(*car_client*, v_i)

Input: T, X_i(internal objects), v_i(insert customer), U(a feasible set of v_i), R_1($v_i \in R_1$), d($v_i \in$Td)

```
1:   U ←
2:   M ←
3:   T ←
4:   for i = 0 ← n do
5:        if (car_client[i].remain > q_i)
6:             M ← M + R_i
7:       end for
8:   for i = d ← 24 do
9:        T ← T + T_i
10:  end for
11:  for i = d ← 24 do
12:     if T_i ∈ T
13:          for j = R_1 ← R_n do
14:              if (R_i ∈ M)
15:                  for k = 0 ← |R_j | do
16:                      v_k ← car_client[j].route[k].number
17:                         if (a_{k-1} < l_k)
18:                              a_k ← a_{k-1} + s_{k-1} + t_{(k-1,k)}
19:                         end if
20:                         if (a_k < l_k)
21:                           F_jk ← min_{i≤1<N} {l_i − (D_i + Σ_{i≤p<1} t_{p,p+1})}
22:                          end if
23:                           if (a'_{k+1} − a_{k+1} < F_jk)
24:                               U ← U + {v_k}
25:                           end if
26:                  end for
27:               end if
28:       end for
29:   end if
30: end for
```

Output: A feasible set

3.3.4 Rules in Cell 4: Mutation Operation

In subsection, the paper lists the pseudo code of IUOX in order to clearly describes this operation. Because *segment_point* crossover need to a feasible set U storing all inserted position of a client in executing IUOX, another function *Swap_po* such as ALGORITHM 4 is proposed.

Definition for mutation operator.

Definition 3: *A node v_1 is selected randomly from a route R_1 obtained from objects X_i ($X_i \subseteq G$), and then compute a swap set U by using a technique satisfying time windows, and randomly select another node v_2 ($v_2 \in R_2$ & $R_2 \subseteq X_i$) from U, and then execute $R_1 = R_1 - \{v_1\}$ and $R_2 = R_2 - \{v_2\}$, R_1 and R_2 are recomputed arrival time of each vi in these two routes after deleting nodes v_1 and v_2. Finally, execute two operations $R'_1 = R_1 + \{v_2\}$ and $R'_2 = R_2 + \{v_1\}$, obtaining a new object X'_i.*

where $i = 0, 1, 2, ..., S-1$, let a route R_1 denote $R_1 = (v_k, ..., v_1, ..., v_{lt})$, $R_1 = R_1 - \{v_1\}$ is that a node v_1 is deleted from current route R_1, obtaining $R_1 = (v_k,...,v_{lt})$, $R'_1 = R_1 + \{v_2\}$ is that a node v_2 is respectively reinserted to the position of v_1 in route R_1, obtaining $R'_1 = (v_k, ..., v_2, ..., v_{lt})$, the operation of R_2 is similar to that of R_1, v_k is an initial node of R_1 and l_t is the length of a route

The pseudo codes for mutation operator. In executing mutation operation, we need to use function *swap_op* to obtain an exchange-feasible set U of a node v_k ($k \in [1, N]$). According to the mutated characteristic of objects, the pseudo codes of computing U are as ALGORITHM 1.

ALGORITHM 5 mainly describes three parts: (1) sequences 4~7 record route set M satisfying max load of vehicles; (2) sequences 8~10 computes a classed set T n of all clients according to their windows; (3) sequences 11~30contained three loops: the first loop takes R_1 ($R_1 \subset M$)from 1 to 24, the second loop takes T_1 ($T_1 \subset T_n$) from 1 to n, and the nested loop takes clients v_1 from1 to *car_client*[R_t].*route.size*. For the nested loop, firstly, adjust whether the latest service time l_k is less than arrival time a_{k-1} of v_k: if meet, adjust whether increment of arrival time $a'_{k+1}$is less than delay time F_{Rt}: if less than, then add v_k to set U; else, discard and continue until all clients are explored, obtaining set U.

In the optimization phrase, we have used an improved mutation rule to increase the diversity of objects. The single module of mutation operation be introduced as ALGORITHM 5.

ALGORITHM 5 Mutation: New solution generation

Input: X_i (an object), R_1, R_2 (two routes), v_1, v_2, U (feasible sets)

1: $F_1 \leftarrow$
2: a = rand()/|X_i|
3: v_1 = rand()/|Ra|
4: $U \leftarrow$ *Swap_po*(X_i, v_1)
5: k = rand()/|U|
6: If (v_2 satisfies the condition of inserting the position of v_1)
7: $R_1 = R_1 - v_1$
8: $R_2 = R_2 - v2$
9: $R'_1 \leftarrow R_1 + \{v_2\}$
11: $R'_2 \leftarrow R_2 + \{v_1\}$
12: $d_{R1} = d_{R1} - d_{v1} + d_{v2}$
13: $d_{R2} = d_{R2} - d_{v2} + d_{v1}$
14: for i = 0 $\leftarrow$ |R_1| do
15: $a_i = a_{i-1} + \max(d_{i-1}, i, e_i) + s_i$
16: for i = 0 $\leftarrow$ |R_2 | do
17: $a_j = a_i - 1 + \max(d_{j-1}, i, e_j) + s_j$
18: else v_2 is selected from U

Output: A new object X'_1

ALGORITHM 5 mainly introduces mutation process of objects. There are the following steps: (a) a node v_1 is randomly selected from an object X_i such as steps 2-3; (c) an exchanged set U of v_1 is computed by calling ALGORITHM 1 and randomly select a node v_2($v_2 \in R_2$) from set U; (d) adjusting whether v_2 satisfies the condition of exchanging with v_1: if meet, exchange v_1 with v_2 and update vehicle surplus and arrival time of each node of two routes; else, return above steps until choosing a feasible node, obtaining a new object X'_1;

Evolution rules of mutation operator in the cell. According to the above introduce of mutation operation, the rule of executing is designed as the following:

$$r_5 = \{(x,y)_{\neg(v_m, v_n)} \rightarrow (z,w)_{\neg(v_n, v_m)} \mid m = 1,2,...,N; n = 1,2,...,N\} \quad \textbf{(16)}$$

where an object $X_i = \{x_{i1}, x_{i2}, ..., x_{i(m-1)}, x_{im}, 0, ..., 0, x_{i(n-1)}, x_{in}, ..., x_{iN}, 0\}$, $x = \{x_{i1}, xi2, xi3, ..., x_{i(k+1)}, ..., x_{i(m-1)}, x_{im}\}$($x \subseteq R_1$ & $R_1 \subseteq X_i$ & $k < m$), $y = \{xi(n-1), x_{in}, ..., x_{iN}\}$($y \subseteq R_2$ & $R_1 \subseteq X_i$)), $z = \{x_{i2}, ..., x_{i(m-1)}, x_{in}\}$, $w = \{x_{i(n-1)}, x_{im}, ..., x_{iN}\}$; v_m is a client on x_{im}, $i = 1, 2, ..., N$.

Executing a rule r_5 needs to the following conditions: (1) the constraint of vehicle capacity: *node*[v_n].*demand* $\leq$ *car_client*[R_1].*remain*; (2) the latest service time is greater than arrival time of previous nodes of x_{im} such as l_n>*node*[v_m−1].*arrive_time*; (3) the earliest time is less than the latest time of a node behind x_{im}, for example, e_n < *node*[v_m+1].*l_time;* (4) the increment of arrival time $\triangle a_{opt}$ ($\triangle a_{opt}$

$= a''_{m+1} - a'_{m+1}$) is less than the minimum delay time F_{nk} such as $\triangle a_{opt} \le F_{nk}$ where $F_{nk} = min_{m\le l<N}\{l_m-(D_m + \sum_{m\le p<1} t_{p,p+1})\}$, a''_{m+1} is new arrival time of v_{m+1} after inserting v_n, a'_{m+1} ($a'_{m+1}\le a''_{m+1}$) is arrival time of v_{m+1} after deleting v_m. In summary, under satisfying above constraints, this operation can increase the diversity of population through gene mutation thus avoiding falling into local optimum and further improving the quality of solutions.

3.3.5 Rules for the Copy in Cells 5 and 6: Elitism Selection Strategy

In this section, we mainly introduce the evolution mechanisms of cells 5 and 6, where cell 5 is used for storing the global object and cell 6 is used for storing some excellent individuals. Because the operator of cell 5 can only be stored, we mainly introduce the process of cell 6 as follows: In order to make some excellent individuals copied directly into next generation, an improved elitism strategy is proposed. This operator is to select an object by using a roulette method according to the following formula.

$$p_i = p_{i-1} + (1 - fit(X_i)/\sum_{i=0}^{S} fit(X_i)) \tag{17}$$

$$fit(X_i) = fit(X_i) \times p_0 \tag{18}$$

where $i\in[0, S-1]$; p_i is the probability of X_i ($X_i \subseteq G$) in the roulette, p_0 ($p_0\in[0.85,0.99]$) is a fair competition coefficient, $fit(X_i)$ is fitness computed by euclidean metric.

Formula (19) expresses a chromosome owned minimum fitness might be selected with greater probability by using a roulette method instead of randomly selecting, which ensures that some optimal chromosomes are pass on to next generation. In order to increase the differences of these individuals, their fitness reduces using formula $fit(X_i) * (1-p_0)$. For cell 6, l_4 objects from cell 2 are selected using sampling with replacement through formula (19) and (20) and send to cell 6. From this we can know, the method can improve the precision of solutions by maintaining optimal individuals in the last generation.

All rules above studied express that they can operate the evolving direction of next generation in searching thus playing a key role for the ordered and successful evolution of this algorithm and further improving the efficiency for quickly searching a global optimum.

3.4 Communication Rules between Cells

Communication mechanisms enabling one cell to influence the behaviour of another one play a fundamental role in multi-cellular organisms where cells have to be able to coordinate their own behaviour for the benefit of the organism as a whole. The further work on rule set $R(R = \{r_1, r_2, ..., r_q\}(|R|=q)$ where an evolving rule r_t ($t < q$) is a matrix rule has been extended by considering four types of design problems for communication as follows:

Type 1. Considering the generated conditions of objects, where each operation-produced membrane cell l (l = 3,4,5) needs to a pre-computed a classed set T for accelerating the evolution of objects which is obtained by calling time classifier. The transportation rule of this cell is designed as follows.

$$r_6 : (T)_{m_1} \to (T,go)_{(m_3\ |m_4\ |m_5)} \tag{19}$$

where $T = T+T_i$, i = 0, 1, ..., 23; $T_i= T_i + \{v_j\}(v_j \in V)$, j = 1, 2, ..., N.

Type 2. Considering a family of objects, $\omega^t_{k,li}= \{X^t_{k1}, X^t_{k2}, ..., X^t_{kli}\}$,where each membrane has designed communication rules R which is obtained according to genetic mechanism. R is a rule set, of the form

$$r_7 : (\omega^t_{k,i})_{m_2} \to (\omega^t_{k,i}, go)_{(m_3\ |m_4\ |m_5)},\ 2 \le k \le 6,\ 0 \le i \le S\text{-}1 \tag{20}$$

where $\omega^t_{k,li}$ stores an object set of the kth membrane at the tth iteration, $R =\{\omega^t_{2,i} \to (\omega^t_{2,i}, go), \omega^t_{3,i} \to (\omega^t_{3,i}, go), ..., \omega^t_{6,i}\to (\omega^t_{6,i}, go)\}$, $t\in[0, t\_max]$.

Type 3. Considering the global update of object set, where a global optimum x^t_{gbest} is updated by local optimum x^t_{lbest} which is computed by the rule of cell 1. This communication rule is as follows.

$$r_8:(x^t_{lbest})_{m_2} \to (x^t_{lbest}, go)_{m_5} \tag{21}$$

where $x^t_{lbest} = min\{fit(X_0), fit(X_2), ..., fit(X_{S-1})\}$, $t \in [0, t\_max]$.

Type 4. Considering the output of optimum x^t_{gbest}, where membrane system send global optimum to the environment cell 0. The corresponding rule is designed as follows.

$$r_9 : x^t_{gbest} \rightarrow (x^t_{gbest}, out) \quad \textbf{(22)}$$

where $x^t_{gbest} = \min\{x^0_{lbest}, x^1_{lbest}, ..., x^t_{lbest}\}$.

To take care of the priority relation we modify the first micro-step as follows:an object can be assigned to a rule only if no object can be assigned to a rule of a higher priority. Hence we have a competition for rule application and not a competition for choosing perhaps the same objects. In the paper, the characteristics of evolution mechanism makes the rules owning the priority for types 1~4 such as $r_6 > r_7$, $r_7 > r_8$, and $r_8 > r_9$ where r_7 contains different sub-rules r_{2_3}, r_{2_4}, r_{2_5} and r_{2_6} where four sub-rules have no priority, which ensures that the whole system can orderly work and make objects evolve successfully.

3.5 Termination Condition and Output

In this subsection, the paper designs termination conditions of membrane computing and output domain. The detailed introductions are as steps a. and b.

a. For MGA, in order to ensure the convergence of the algorithm, we design the following termination criterions:

(a) The program has achieved termination states when $x_{gbest} = x_0$ where x_0 is an optimum of database;

(b) The algorithm can stop when the difference ratio between x_{gbest} and x_0 remains unchanged in multiple iterations where $ratio = (x_{gbest} - x_0)/x_0 \%100$.

The algorithm terminated immediately just meeting any one of these two conditions.

b. Cell 0 is specified as an output domain receiving the best result x_{gbest}.

4 Application of MGA in VRPTW

According to the characteristics of the model of VRPTW and the designed rules of membrane algorithm, every optimal route is expressed by an evolved object which is computed by membrane algorithm. The detailed steps in solving VRPTW are as follows.

Step 1: Construct membrane system; Initialize population; t=0;

Step 2: In cell 1, all clients are classed using the rule of this cell and stored in a classed set T;

Step 3: In tth iteration, cell 2 produces S objects by using the computational process of this cell, and compute x_{lbest} and x_{gbest}.

Step 4: Cell 3 receives l_1 objects of cell 2 selected by formula (19) using the rule r_7, and then execute rules r_1 and r_2, obtaining l_1 crossover objects under the action of r_6.

Step 5: Cell 4 receives l_2 objects of cell 2 using the rule r_7, and execute rules r_3 and r_4, obtaining l_2 crossover objects under the rule r_6.

Step 6: Cell 5 receives l_3 objects using the rule r_7, and execute rule r_5, obtaining l_3 objects under the constraints of the rule r_6.

Step 7: Cell 6 receives l_4 objects selected by formulas (19)-(20) by using l_1, obtaining l 4 optimal objects under the rule r_6.

Step 8: S produced objects of cell 2 are transported to outside environment, and cell 2 receives S new objects from cell 3~6 again and update x_{gbest} by using r_8, t++.

Step 9: If $t < t\_max$, then output x_{gbest} using r_9; else, turn Step 3.

5 Experiments and Results

We coded the above procedure using Visual C++, and ran the program 30times on different randomly generated seeds for each data set. The data sets chosen were Solomon's 50 and 100 customer problems, which is the most population test bed for VRPTW created by Solomon in 1987. These are 6 types of problems,C1, R1, RC1, C2, R2 and RC1. "C" problems are clustered problems, "R" problems are randomly generated and 'RC' problems are a combination of the two. The problems numbered "1" are based on narrow time windows and problems numbered "2" are based on wide time windows. The

computer used had 2.30GHzprocessors. For experimental results, Table 1 lists the range of parameters tested and its values; Table 2 lists the parameter values of the algorithms to be compared; Table 3 list the compared results of 100 clients; Table 4 gives the compared results between MGA and other heuristics. The detailed design is described as follows.

Table 1. Nomenclature for parameters of membrane algorithm and differential evolution values used for both the special and stochastic optimization trials

Stage	Parameter name	Tested name	Parameter optimal values					
			C-50	R-50	RC50	C100	R100	RC100
Population size	τ	[50;2500]	1000	1000	1000	500	500	500
Iteration times	σ_0	[100;1000]	1500	1500	1500	1000	1000	1000
The classed sets	c_{τ}	[0;24]	24	24	24	24	24	24
External membranes		[0;7]	6	6	6	6	6	6
[a] Cell 3	c_{γ}	[0;0.6]	750	750	750	228	228	228
[b] Cell 4	c_{α}	[0;0.2]	50	50	50	122	122	122
[c] Cell 5	c_{β}	[0;0.15]	150	150	150	125	125	125
[d] Cell 6	c_{κ}	[0;0.05]	50	50	50	25	25	25
Fair weight coefficient	f_{φ}	[0;0.99]	0.90	0.90	0.90	0.99	0.99	0.99
Running times	r_{ρ}	[0;100]	50	50	50	30	30	30

Table 2. Tabu search, genetic algorithm with simulated annealing mechanism and particle swarm optimization and differential evolution parameters used for both the special and stochastic optimization trials

Parameter	Value
Initial Tabu list length (ρ)	13
Candidate list length (*Max_list*)	23
Maximum number of iterations (σ_1)	500
Number of population (τ)	500
Maximum number of iterations (σ_2)	1000
Crossover probability-extraction customers ($prob^{PC}$)	0.70
Mutation probability-exchange policy ($prob^{PM}$)	0.25
Cloning probability-copy strategy ($prob^{sel}$)	0.05
Annealing cooling iterations ($T = 20$)	3.0
Number of particles (*Num*)	500
Maximum number of iterations (σ_3)	1000
Particle inertia (c_1)	2.05
Particle best inertia (c_2)	2.05
Inertia weight coefficient (w_{max}) -maximum value	0.9
Inertia weight coefficient (w_{min}) -minimum value	0.4

Table 3. The precision comparison for three algorithms under consideration for the proposed algorithm MGA, standard database, and traditional GA specified in the text

Instances	Min		Vehicles	Average		Best	known	Min
	MGA	GA	n	MGA	GA	n	Optimum	Dev%
C101	866	3731.12	11	866	4036.90	10	827.3	4.67
C102	1751	3386.32	12	1924.4	3459.10	10	827.3	111.65
C103	1932	3521.07	14	2212.6	3698.27	10	826.3	133.81
C104	1756	3613.07	15	2119.2	3780.31	10	822.9	113.39
C105	858	3292.40	10	884.8	3387.79	10	827.3	3.71
C201	1363	3013.13	10	1553	3103.10	3	589.1	131.16
C202	1911	3413.63	10	1911	3510.30	3	589.1	224.39
C203	2117	3521.36	11	2359.6	3591.41	3	588.7	259.6
C205	1452	2828.34	10	1578.2	3001.14	3	586.4	147.6
C206	1749	2939.88	10	1929.2	3117.05	3	586.0	198.46
C207	1851	3175.84	10	2077.8	3232.57	3	585.8	215.97
C208	1907	2978.21	10	2094.2	3130.34	3	586.4	252.2

Table 3. The precision comparison for three algorithms under consideration for the proposed algorithm MGA, standard database, and traditional GA specified in the text (continu)

Instances	Min		Vehicles	Average		Best	known	Min
	MGA	GA	n	MGA	GA	n	Optimum	Dev%
R101	1832	3341.71	20	1860	3987.41	20	1637.7	11.86
R102	1681	3287.19	18	1729.6	3758.19	18	1466.6	14.61
R103	1413	3103.31	15	1518.67	3151.90	14	1208.7	16.90
R104	1359	2896.46	14	1371.00	3125.61	11	971.5	39.88
R105	1548	2970.08	15	1569.4	3182.57	-	-	-
R106	1483	2904.71	14	1510	3020.57	13	1234.6	20.12
RC101	1844	4103.41	16	1870.8	4209.17	15	1619.8	13.84
RC102	1651	4038.25	14	1684.2	4137.06	14	1457.4	13.28
RC103	1589	3845.03	13	1637.4	3945.17	11	1258.0	26.3
RC104	1424	3590.48	13	1541.6	3658.93	-	-	-
RC105	1806	3920.08	16	1847.6	4152.23	15	1513.7	19.31
RC107	1447	3475.34	12	1553.2	3706.79	12	1207.8	19.80
RC108	1570	4985.97	14	1642	5028.14	11	1114.2	40.90
RC201	2211	2431.67	9	2234.8	2510.01	9	1261.8	75.22
RC202	2203	2985.04	10	2284.6	3128.69	8	1092.3	101.68
RC208	2040	3115.86	10	2101.6	3261.74	-	-	-
Average	1664.78	3371.74	12.71			9.68	912.58	78.93

Table 4. Performance comparisons for the distance computed by the MGA, Tabu search algorithm (Tabu), genetic algorithm with simulated annealing mechanism (GSAG) and improved particle swarm optimization algorithm (IPSO)

Instances	MGA	Tabu		GSAG		IPSO	
	Distance	Distance	Dev%	Distance	Dev%	Distance	Dev%
C101	866	3643.32	320.71	3868.65	346.73	3812.22	340.21
C105	858	3759.63	338.19	2949.69	243.79	3845.36	348.18
C201	1363	2612.29	91.66	3643.63	167.32	3579.73	162.64
C205	1452	2851.42	96.38	3506.72	141.51	3553.40	144.72
C208	1907	2889.43	51.52	3638.57	90.80	3321.84	74.19
R101	1832	3420.68	86.72	3635.01	98.42	3661.92	99.89
R105	1548	3684.04	137.99	3569.93	130.62	3455.26	123.21
R201	1774	2253.78	27.05	3301.36	86.10	3188.65	79.74
RC101	1844	4352.64	136.04	4565.76	147.60	4773.66	158.88
RC105	1806	4369.73	141.96	4371.44	142.05	4888.14	170.66
RC108	1570	3988.22	154.03	4269.97	171.97	4672.68	197.62
RC201	2211	4255.40	92.46	4126.67	86.64	2222.06	0.50
RC208	2040	3626.45	77.77	4282.75	109.94	4222.38	106.98
Average	1620.85		134.81		151.04	3784.41	154.42

5.1 Results for Parameters Tuning

The following are the parameters of the MGA which were different from GA:

a. Population Generation: Random.

b. The total number of the objects(Population Size), such that:

- The number of objects in cell3 is: c_γ.
- The number of objects in cell4 is: c_α.
- The number of objects in cell5 is: c_β.
- The number of objects in cell6 is: c_κ.

c. Iteration times, where the number of iterations for membranes is the same.

All of the parameters used in MGA are the same as for 50 customers and 100customers, including two crossover rates, mutation rate and their probabilities.When we finish a state transition and get an optimized chromosome from the MGA, cell 2 accepts all objects of the above four membranes and then sends them to each membrane under the premise of the above parameters after selecting by using the

roulette method. However, in compared algorithm, there is only one crossover operation, mutation and cloning. And for its parameters, they are obtained by using the parameter results of GA in literature [59] such that population size is 1000, iteration size is 1500, crossover rate is 0.80, mutation rate is 0.15 and copying rate is 0.05, all operations of traditional algorithm are doing in the same conditions as the above parameters. The comparison between two algorithms are done based on the above parameters. In the experimental phase, multiple combinations of parameters shown in Table 1 and Table 2 has been verified. For MGA, the testing ranges of parameters, as listed in the above tables, have been set with necessary margin and thoroughly analyzed using different instances.

Table 2 summarizes the parameters used for the tabu search algorithm, genetic algorithm with annealing mechanism and particle swarm optimization. It is noted that these parameters are obtained by constantly testing all algorithms of Table 4.

5.2 The Comparison of the Precision with Other Algorithms

In order to compare the precision of MGA with other algorithms using the same problems, we design a series of the experiments. For the test of the in-stances, the paper divided into two types from the windows. The results are shown in table 3. For this table, the first column is the problem name, and the column 2 is the value of the best-obtained solution (MGA), along with the results (column 3) of the tradition of the heuristic, that first computed these results. In column 4, n stands for vehicle number; The column 5 and column 6, the comparison of the average between two algorithms. In columns 7-8, the tested vehicle number and the obtained optimum of standard database. In the last column we have shown the values attained from the form of Dev(%) = (Min MGA − Min GA)/Min MGA × 100%. The experimental results are listed as tables 4 and 5.

The experimental results of Table 3 shows that MGA has attained better solutions compared with traditional algorithm shown in column 2. For the best solutions, this is because the function of membranes and node classifier plays a key role for improving the performance of the algorithm, where the membranes expand the diversity of population and the communication between membranes makes it evolving towards a optimal direction thus making the algorithm obtaining some better objects. In column 4, we list the required the number of vehicles, the comparative results shows that our algorithm can service all customers by using a very lower the number of vehicles that have almost equaled to the best values besides these instances C201 C208, for better results, this is because this improved crossover operator obtains some feasible solutions by constantly inserting into a single node thus ensuring that the vehicles reduces in the process of computing; for worse results, this is because the limitation of time windows leads to the high vehicles in solving VRPTW. In the last column,we give a deviation ratio of the minimum of MGA and standard database. For the computational values, we can know that, compared with the best known of standard database, in twenty-eight instances, the ratios of two instances are controlled within 10 percentage, those of five instances are within 15 percent-age, and that of five instances can also be in 20%, other instances are controlled different percentage, these results shows that the improved strategies plays a key role in improving the performance of the algorithm. According to comprehensive analysis, the performance of the algorithm can get improved for solving VRPTW.

5.3 Performance Comparisons for the Distance Optimization Using the MGA and the Heuristic Algorithms

In order to verify the performance of membrane algorithm, the paper gives different comparisons with other algorithms in Table 4. The table mainly compares the precision of solutions of MGA, Tabu, GSAG and IPSO algorithms. For four algorithms, their running parameters are listed as Table 2. Each algorithm is executed under the conditions of tuning the best status. According to the solving method of four algorithms, the experimental results of these heuristics are listed as Table 4.

It is apparent from Table 4 that the MGA algorithm consistently outperform the algorithms that uses tabu search algorithm, GSGA and IPSO, with an average decrease in present value of 134.81%, 151.04% and 115.42%, respectively. For better solutions of membrane algorithm, this is because we introduce membrane structure in order to make the algorithm search towards different space thus increasing the diversity of population and further enhancing the precision of solutions, and an improved roulette mechanism to make the algorithm evolve towards the direction of the optimal solutions thus further improving the precision of solutions; In order to make the algorithm enhance the successful probability

of generating feasible objects, the paper introduces an improve crossover opera-tor, according to the computational results, these solutions can be accepted inthe case of improving this crossover operator. In summary, these different improvements can really improve the quality of solutions in a certain extent, which shows the effectiveness of designing through membrane algorithm.

6 Conclusions and Remarks

Considering the efficiency about NP-hard optimization problems, a recently proposed membrane algorithm is used to try to solve this problem which is an optimization algorithm that genetic mechanism is embedded in membrane computing based on bionic cell. In the algorithm, we introduce a node classifier mechanism in a cell in order to improve the computational efficiency; we introduce an improved crossover operator in a cell so as to increase the successful probability of crossover operator; we introduce a mutation operator in a cell to expand the search space; The communication rule between cells are designed in order to exchange information each other. The experimental results show that the algorithm has some advantages in solving the optimization problems compared with other literature or traditional one.

The paper incorporates the character of cell-like P systems and genetic operation to solve the VRPTW problem. There are many kinds of membrane systems have been proposed and investigated, which can also been considered to solve the VRPTW problem. Such as for the character of numerical, numerical P systems [60-62] maybe have the potential convenience in optimizing the VRPTW.

Acknowledgments

The work was supported by National Natural Science Foundation of China (61179032, 61502012), the Special Scientific Research Fund of Food Public Welfare Profession of China (201513004-3), the Guiding Scientific Research Project of Hubei Provincial Education Department (2017078), the Humanities and Social Sciences Fund Project of Hubei Provincial Education Department (17Y071).

References

[1] J. Lenstra, K.A. Rinnooy, Complexity of vehicle routing scheduling problems, Networks 11(1981) 221-227.

[2] A. Haghani, S. Jung, A dynamic vehicle routing problem with time-dependent travel times, Computers & Operations Research 32(11)(2005) 2959-2986.

[3] A. Kok, E. Hans, J. Schutten, Vehicle routing under time-dependent travel times: the impact of congestion avoidance, Computers & Operations Research 39(5)(2012) 910-918.

[4] J. Nalepa, M. Blocho, Adaptive memetic algorithm for minimizing distance inthe vehicle routing problem with time windows, Applied Soft Computing 20(6)(2016) 2309-2327.

[5] B. Melián-Batista, A. Santiago, F. AngelBello, A bi-objective vehicle routing problem with time windows: a real case in tenerife, Applied Soft Computing 17(2014) 140-152.

[6] D. Tas, N. Dellaert, T. Van, The time-dependent vehicle routing problem with soft time windows and stochastic travel times, Transportation Research PartC: Emerging Technologies 48(2014) 66-83.

[7] H. Florent, F. Dominique, G. Rodolphe, N. Olivier, Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows, European Journal of Operational Research 2(2016) 551-559.

[8] L. Sumaiya, M. Kaykobad, M. Rahman, Solving the multi-objective vehicle routing problem with soft time windows with the help of bees, Swarm and Evolutionary Computation 24(2015) 50-64.

[9] J. Brito, F. Martínez, J. Moreno, J. Verdegay, An ACO Hybrid metaheuristic forClose open vehicle routing problems with

time windows and fuzzy constraints, Applied Soft Computing 32 (2015) 154-163.

[10] K. Ghoseiri, S. Ghannadpour, A hybrid genetic algorithm for multi-depot homogenous locomotive assignment with time windows, Applied Soft Computing 10(1)(2010) 53-65.

[11] Z. Ursani, D. Essam, D. Cornforth, R. Stocker, Localized genetic algorithm for vehicle routing problem with time windows, Applied Soft Computing 11(8)(2011) 5375-5390.

[12] Gh. Păun, Computing with membranes, Journal of Computer System Science 61(1)(2000) 108-143.

[13] C. Martín-Vide, Gh. Păun, J. Pazos, A. RodríGuez-Patón, Tissue P systems, Theoretical Computer Science 296(2)(2003) 295-326.

[14] M. Lonescu, Gh. Păun, T. Yokomori, Spiking neural P systems, Fundamenta Informaticae 71(2-3)(2006) 279-308.

[15] M.J. Pérez-Jiménez, A. Núnez, Solving The subset-sum problem by P systems with active membranes, New Generation Computing 23(4)(2005) 339-356.

[16] C. Ferretti, G. Mauri, Gh. Păun, On three variants of P systems with string-objects, in: Proc. Pre-Proceedings of Workshop on Membrane Computing, 2001.

[17] X. Zhang, L. Pan, A. Păun, On universality of Axon P systems, IEEE Transactions on Neural Networks and Learning Systems 26(11)(2015) 2816-2829.

[18] F. Cabarle, H. Adorna, M.J. Pérez-Jiménez, T. Song, Spiking neural P systems with structural plasticity, Neural Computing and Applications 26(8)(2015) 1905-1917.

[19] T. Song, L. Pan, Spiking neural P systems with request rules, Neurocomputing 193(2016) 193-200.

[20] L. Pan, Gh. Păun, Spiking neural P systems with anti-spikes, International Journal of Computers Communications & Control 4(3)(2009) 273-282.

[21] T. Song, L. Pan, J. Wang, I. Venkat, K. Subramanian, R. Abdullah, Normal forms of spiking neural P systems with anti-spikes, IEEE Transactions on NanoBio-science 11(4)(2012) 352-359.

[22] X. Zeng, L. Xu, X. Liu, L. Pan, On languages generated by spiking neural P systems with weights, Information Sciences, 278(2014) 423-433.

[23] X. Zeng, X. Zhang, T. Song, L. Pan, Spiking neural P systems with thresholds, Neural computation, 26(7)(2014) 1340-1361.

[24] T. Song, L. Pan, Gh. Păun, spiking neural P systems with rules on synapses, Theoretical Computer Science 529(2014) 82-95.

[25] X. Zhang, X. Zeng, L. Pan, Weighted spiking neural P systems with rules on synapses, Fundamenta Informaticae 134(1)(2014) 201-218.

[26] T. Song, X. Liu, Y. Zhao, X. Zhang, Spiking neural P systems with white hole neurons, IEEE Trans on Nanobioscience 15(7)(2016) 666-673. DOI: 10.1109/TNB.2016.2598879

[27] T. Song, P. Zheng, D. Wong, X. Wang, Design of logic gates using spiking neural P systems with homoge-neous neurons and astrocytes-like control, Information Sciences 372(2016) 380-391.

[28] T. Wu, Z. Zhang, Gh. Păun, L. Pan, Cell-like spiking neural P systems, Theoretical Computer Science 623(2016) 180-189.

[29] G. Ciobanu, L. Pan, Gh. Păun, M.J. Pérez-Jiménez, P systems with minimal parallelism, Theoretical Computer Science 378(1)(2007) 117-130.

[30] B. Song, L. Pan, Computational efficiency and universality of timed P systems with active membranes, Theoretical Computer Science 567(2015) 74-86.

[31] B. Song, M.J. Pérez-Jiménez, L. Pan, An efficient time-free solution to sat problem by P systems with pro-teins on membranes. Journal of Computer and System Sciences 82(2016) 1090-1099.

[32] T. Song, L. Pan, Gh. Păun, Asynchronous spiking neural P systems with local synchronization, Information Sciences 219(2013) 197-207.

[33] B. Song, L. Pan, Flat maximal parallelism in P systems with promoters, Theoretical Computer Science 623(2016) 83-91.

[34] T. Song, L. Pan, K. Jiang, B. Song, W. Chen, Normal forms for some classes of sequential spiking neural P systems, IEEE Transactions on NanoBioscience 12(3)(2013) 255-264.

[35] T. Song, L. Pan, Spiking neural P systems with request rules, Neurocomputing 193(2016) 193-200.

[36] B. Song, L. Pan, The computational power of tissue-like P systems with promoters, Theoretical Computer Science 641(2016) 43-52.

[37] B. Song, L. Pan, M.J. Pérez-Jiménez, Tissue P systems with protein on cells, Fundamenta Informaticae 144(2016) 77-107.

[38] T. Song, P. Zheng, D. Wong, X. Wang, Design of logic gates using spiking neural P systems with homogeneous neurons and astrocytes-like control, Information Sciences 372(2016) 380-391.

[39] B. Song, M.J. Pérez-Jiménez, L. Pan, Efficient solutions to hard computational problems by P systems with sym-port/antiport rules and membrane division, BioSystems 130(2015) 51-58.

[40] B. Song, M.J. Pérez-Jiménez, L. Pan, Computational efficiency and universality of timed P systems with membrane creation, Applied Soft Computing 19(11)(2015) 3043-3053.

[41] B. Song, M.J. Pérez-Jiménez, L. Pan, An efficient time-free solution to sat problem by P systems with proteins on membranes, Journal of Computer and System Sciences 82(2016) 1090-1099.

[42] Y. Niu, I. Venkat, A. Khader, K. Subramanian, Uniform solution to common algorithmic problem by P systems working in the minimally parallel mode, Fundamenta Informaticae 136(3)(2015) 285-296.

[43] Y. Niu, J. Xiao, Y. Jiang, Solving 3-coloring problem with time-free tissue P systems, Chinese Journal of Electronics 25(3)(2016) 407-412.

[44] G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez (eds.), Applications of Membrane Computing, Springer, Berlin, 2006.

[45] G. Zhang, J. Cheng, T. Wang, X. Wang, J. Zhu, Membrane Computing: Theory and Applications, Science Press, Beijing, China, 2015.

[46] D. Díaz-Pernil, A. Berciano, F. Peña-Cantillana, M. Gutiérrez-Naranjo, Segmenting images with gradient-based edge detection using membrane computing, Pattern Recognition Letters 34(8)(2013) 846-855.

[47] D. Díaz-Pernil, F. Peña-Cantillana, M. Gutiérrez-Naranjo, A parallel algorithm for skeletonizing images by using spiking neural P systems, Neurocomputing 115(2013) 81-91.

[48] X. Liu, Z. Li, J. Liu, L. Liu, X. Zeng, Implementation of arithmetic operations with time-free spiking neural P systems, IEEE Transactions on NanoBioscience 14(6)(2015) 617-624.

[49] G. Zhang, H. Rong, Z. Ou, M.J. Pérez-Jiménez, M. Gheorghe, Automatic design of deterministic and non-halting membrane systems by tuning syntactical ingredients, IEEE Transactions on NanoBioscience 13(3)(2014) 363-371.

[50] H. Peng, J. Wang, M.J. Pérez-Jiménez, H. Wang, J. Shao, T. Wang, Fuzzy reasoning spiking neural P system for fault diagnosis, Information Sciences 235(2013) 106-116.

[51] J. Wang, P. Shi, H. Peng, Membrane computing model for IIR filter design, Information Sciences 329(2016) 164-176.

[52] T. Wang, G. Zhang, J. Zhao, Z. He, J. Wang, M.J. Pérez-Jiménez, Fault diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems, IEEE Transactions on Power Systems 30(3)(2015) 1182-1194.

[53] G. Zhang, J. Cheng, M. Gheorghe, A membrane-inspired approximate algorithm for traveling salesman problems, Romanian Journal of Information Science and Technology 14(1)(2011) 3-19.

[54] Y. Niu, S. Wang, J. He, J. Xiao, A novel membrane algorithm for capacitated vehicle routing problem, Applied Soft Computing 19(2)(2015) 471-482.

[55] J. He, J. Xiao, X. Liu, T. Wu, T. Song, A novel membrane-inspired algorithm for optimizing solid waste transportation, Optik- International Journal for Light and Electron Optics 126(23)(2015) 3883-3888.

[56] G. Zhang, C. Liu, M. Gheorghe, Diversity and convergence analysis of membrane algorithm, in: Proc. Bio-Inspired Computing: Theories and Applications (BIC-TA), 2010 IEEE Fifth International Conference, 2010.

[57] X. Zhang, B. Wang, Z. Ding, Implementation of membrane algorithms on GPU, Journal of Applied Mathematics (2014), DOI: 10.1155/2014.307617

[58] T. Nishida, An application of P-system: a new algorithm for NP-complete optimization problems, Cybernetics and Informatics 5(2004) 109-112.

[59] Y. Duan, K. Zhou, W. Dong, Improved genetic algorithm to optimization pattern in traffic network layout problem, Communications in Computer and Information Science 562(2015) 138-149.

[60] Z. Zhang, L. Pan, Numerical P systems with thresholds, International Journal of Computers Communications & Control, 11(2)(2016) 292-304.

[61] Z. Zhang, T. Wu, L. Pan, Gh. Păun, On string languages generated by sequential numerical P systems, Fun-damenta Informaticae 145(4)(2016) 485-509.

[62] Z. Zhang, T. Wu, A. Păun, L. Pan, Numerical P systems with migrating variables, Theoretical Computer Science. 641(2016) 85-108.