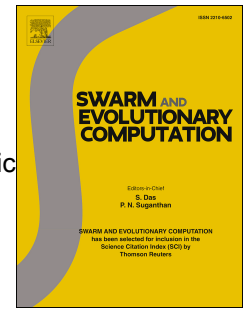
Accepted Manuscript



A self-adaptive evolutionary algorithm for dynamic vehicle routing problems with traffic congestion

Nasser R. Sabar, Ashish Bhaskar, Edward Chung, Ayad Turky, Andy Song

PII: S2210-6502(18)30340-7

DOI: https://doi.org/10.1016/j.swevo.2018.10.015

Reference: SWEVO 464

To appear in: *Swarm and Evolutionary Computation BASE DATA*

Received Date: 26 April 2018

Revised Date: 23 October 2018

Accepted Date: 29 October 2018

Please cite this article as: N.R. Sabar, A. Bhaskar, E. Chung, A. Turky, A. Song, A self-adaptive evolutionary algorithm for dynamic vehicle routing problems with traffic congestion, *Swarm and Evolutionary Computation BASE DATA* (2018), doi: https://doi.org/10.1016/j.swevo.2018.10.015.

A Self-adaptive Evolutionary Algorithm for Dynamic Vehicle Routing Problems with Traffic Congestion

Nasser R. Sabar [1], Ashish Bhaskar [2], Edward Chung [3], Ayad Turky [4], Andy Song [4]

[1] *Department of Computer Science and Information Technology, La Trobe University, Melbourne, VIC 3083, Australia*

[2] *Smart Transport Research Centre, Queensland University of Technology, Brisbane, QLD 4000, Australia*

[3] *Department of Electrical Engineering, The Hong Kong Polytechnic University, Hung Hom, Hong Kong*

[4] *School of Computer Science and Information Technology, RMIT University, Melbourne, VIC 3000 Australia*

Abstract

The Dynamic Vehicle Routing Problem (DVRP) is a complex variation of classical Vehicle Routing Problem (VRP). The aim of DVRP is to find a set of routes to serve multiple customers at minimal total travelling cost while the travelling time between point to point may vary during the process because of factors like traffic congestion. To effectively handle DVRP, a good algorithm should be able to adjust itself to the changes and continuously search for the best solution under dynamic environments. Because of this dynamic nature of DVRP, evolutionary algorithms (EAs) appear highly appropriate for DVRP as they search in a parallel manner with a population of solutions. Solutions scattered over the search space can better capture the dynamic changes. Solutions for new changes are not built from scratch as they can inherit problem-specific knowledge from parent solutions. However, the performance of EA is highly dependent on the utilised configura-

Email addresses: n.sabar@latrobe.edu.au (Nasser R. Sabar [1]), ashish.bhaskar@qut.edu.au (Ashish Bhaskar [2]), edward.cs.chung@polyu.edu.hk (Edward Chung [3]), ayad.turky@rmit.edu.au (Ayad Turky [4]), andy.song@rmit.edu.au (Andy Song [4])

tion. To address this issue, we propose a self-adaptive EA for DVRP. The proposed EA evolves a set of configurations including parameter values, operator types, combination of operators and order of operator invocation. The configurations are then encoded into DVRP solutions. So the search can use different configuration during a search process to effectively handle the dynamic changes and guide the search process towards promising areas. Two well known routing problems with traffic congestion, vehicle routing and the travelling salesman, were used to evaluate the performance of the proposed EA. The results demonstrate that under same conditions on both problems the proposed self-adaptive EA is better than standard EA and other algorithms from literature.

Keywords: dynamic optimisation, meta-heuristics, vehicle routing

1. Introduction

Vehicle routing is obviously significant in the areas of logistics, transportation and related industries. Thus it has attracted a large number of researchers and practitioners from various sectors to study it, which is often referred as VRP (Vehicle Routing Problem). Classical VRP is a well-known combinatorial optimization problem and has been intensively investigated [5], [44]. VRP involves a set of customers geographically distributed at different locations and a fleet of vehicles. The goal is serving all customers at minimal cost (e.g. travelling distance, time, fuel etc.) while respecting all constraints. Due to the nature of real-world vehicle routing problems, several variants of VRP have been formulated. Each of these VRP variant accommodates certain constraints and factors to reflect one type of real-life scenarios [44]. Well-known VRP variants include: (i) CVRP, the Capacitated Vehicle Routing Problem which has vehicle capacity as a hard constraint [44]; (ii) VRPTW, the Vehicle Routing Problem with Time Windows where each customer can only accept services/deliveries during a predefined time window [2]; (iii) VRPMT, the Vehicle Routing Problem with Multiple Trips, where a vehicle can take more than one trip/task [46]. In these vehicle routing scenarios, the travel time from one point to another is a constant.

Classical VRP and its variants are known as $\mathcal{NP}$-hard in terms of the problem complexity [46]. That means finding the optimal solution could be impractical for VRP instance of reasonable size due to the prohibitive computational resource required. Exact methods, which guarantee optimal

solutions, are only advisable to be used on small instances [44]. In reality small instances have little practical values as real world problems often are large in size. Thus, meta-heuristic algorithms are better alternatives in these scenarios, as they can often generate solutions of good quality within an acceptable amount of time. This kind of method offers no guarantee of optimality but high application value as the good solutions generated by them are often not far from the optimal solutions [44]. Typical meta-heuristic algorithms include tabu search [11], simulated annealing [30], evolutionary algorithms [1], ant colony [10] and variable neighbourhood algorithms [15].

In reality most of logistics and transportation problems are dynamic by nature [31]. Only limited information is available at the beginning of a trip. New information arrives over time. For example, a new order from customer may appear whilst vehicles are already on road serving customers [27]. Another important factor is traffic condition, which may vary dramatically in different time of a same day, or a same time in different days. It is difficult to foretell the level of traffic congestion [31]. This leads to a challenging and realistic dynamic variant of the VRP, which is denoted as DVRP, referring the vehicle routing problem with traffic congestion. Dynamic order occurs much less often in comparison. In DVRP the exact travel time between customers is not known in advance and subject to the level of the traffic congestion on the path, meaning that after generating a set routing plans and after the vehicles have left the depot to serve customers, the travel time between customers may change. When a change occurs, the optimisation algorithm should adapt to it and find new solutions at minimal cost. However, DVRP is relatively unexplored despite its theoretical importance and practical values.

Similar to traditional VRP, DVRP is also an $\mathcal{NP}$-hard problem [21], hence meta-heuristic methods are suitable.An effective way for solving DVRP with meta-heuristic algorithms is to utilise evolutionary algorithms (EAs). The major characteristic of EAs is that they operate on a population of solutions. Hence the search can have multiple fronts scattered over the solution space. This is particularly beneficial in order to capture the dynamic changes in DVRP. Another benefit of an EA is that it can transfer problem-specific knowledge from one generation to another, which is very helpful in dealing with cyclic dynamic environment [7]. EA has also shown a remarkable success in solving dynamic and noisy optimisation [4],[33], [29], [45].

However, the performance of an EA is highly dependent on the configuration including the parameter settings and the utilised operators [6]. A fixed configuration may lead to a bias towards a certain area of the search

space resulting in sub-optimal results, especially for dynamic problems such as DVRP. Over the years, several attempts have been made on controlling and adapting EA configurations such as parameter control, operator selection and self-adaptive methods. Parameter control methods seek for the appropriate values in off-line manner or during the course of evolutionary search [40],[6]. Operator selection methods seek for the suitable operators such as mutation and crossover on the fly [9]. Self-adaptive methods encode EA configuration, including parameters or operators, along with the solutions. So the search is on the solution as well as on the best configuration during the entire process [41]. These existing adaptive configuration methods have shown improvement compared to fixed configuration approaches. However they focus on one or a few parameters or operators hence are not suitable for tuning a large set of operators and parameters. In addition these methods do not consider complex configurations such as combining parameters and operators.

Therefore we propose a self-adaptive EA in this work to handle DVRP more effectively. In addition to the aforementioned limitations of the existing adaptive EAs such as [40], [41] , the sequence of applying operators is also not considered in these methods. Furthermore solutions of the same population in these methods have identical configurations which is not necessarily the best strategy. In DVRP, it is not known in advance which operator should be applied first on which solution. The utilisation of operators including the sequence should be adaptive to the search to escape from local optima and to find high quality solutions more effectively. For example the mutation operator should be applied before crossover operator. But in another case the order should be reversed. So the convergence and diversity, which are the two critical factors in any evolutionary search process, can be better maintained.

Our proposed algorithm is designed for complex configurations including the proper parameter values, the choice of operators, the combination of operators and parameters, and the sequence of applying these operators. The configurations are encoded as a part of an individual solution and participate the evolutionary search process. It should be noted that the proposed algorithm falls under the definition of memetic computing paradigm [28], [35],[37], [38], [39].

The study uses the DVRP benchmark released by Mavrovouniotis and Yang [22]. Traffic factors are introduced in this benchmark. The travel times between all customers is affected by this traffic factors, which change

constantly depending on the time and the day. Along with the benchmark Mavrovouniotis and Yang proposed several variants of ant colony optimization algorithms for solving DVRP. Various memory schemes are integrated in their algorithms. Their study shows that the diversity of solutions plays a big role in guiding the search to adapt dynamic changes. On the other hand their study shows that the performance of ant colony algorithms was highly dependent on settings such as these memory schemes. This highlights the need for our study on self-adaptive method.

The main contributions of this work can be summarised as follows:

- an evolutionary algorithm that uses variable parameter values and customised operators is proposed to solve DVRP more effectively.
- a self-adaptive evolutionary algorithm that evolves various configurations of operators and parameters.
- a self-adaptive mechanism is presented which is capable of discovering good combinations of parameters and operators.
- a self-adaptive evolutionary algorithm that searches for the proper parameter values, operator types, combination of operators and operators application sequence.
- a set of methods have been evaluated on benchmark datasets and compared with state-of-the-art methods from the literature. Good performance has been demonstrated.

The rest of the paper is organised as follows. Section 2 describes the classical VRP and DVRP. The proposed algorithm for DVRP is presented in Section 3. Section 4 shows the experimental setup, including the benchmark instances, the preparation and the performance evaluation. The experimental results are presented and discussed in Section 5, including effectiveness evaluation and comparisons with results from the literature. The conclusions is given at Section 6.

2. Problem description

In this section, we first formally describe the classical VRP, then the dynamic VRP variant.

2.1. Classical Vehicle Routing Problems

In classical VRP, there is a set of geographically spreaded customers with known demands and a fleet of vehicles of fixed capacity. VRP can be formulated as a mathematical model as follows. Let $G(V, E)$ be a complete directed graph where $V = \{v_0, v_1, \ldots, v_n\}$ is a set of nodes. Node v_0 is the depot which has m vehicles and nodes $v_1 \ldots, v_n$ represent a set of customers. Vehicles have identical capacity, $Q = Q_1, \ldots, Q_m$. E represents a set of edges connecting customers v_i and v_j, $E_{ij} = \{(v_i, v_j) : v_i, v_j \in V, i < j\}$. Each edge E_{ij} has a non-negative value which is the cost e.g. the travel time between v_i and v_j. The cost is defined by a matrix $C = (c_{ij})$. An entry c_{ij} of the matrix C represents the cost between customers v_i and v_j. Each customer v_i is associated with a non-negative value representing q_i goods to be delivered or picked at this customer. Each delivery has a service time δ_i. The goal of DVRP optimisation is to find a set of vehicle routes to serve all customers at minimal cost while satisfying the following constraints:

- All vehicles must start and end their routes, $R_1, \ldots, R_m$, at the depot v_0.
- The total demand assigned for each vehicle should not exceed the vehicle capacity.
- Each customer is visited only once in the delivery plan.
- The total duration of each route should not exceed the given global upper bound.

The cost, C, of each route $R_i = \{v_{\pi,i(0)}, v_{\pi,i(1)}, \ldots, v_{\pi,i(n+1)}\}$ of vehicle i is calculated as follows:

$$C(R_i) = \sum_{j=0}^{n_i} c_{\pi(j),\pi(j+1)} \tag{1}$$

where $v_{\pi(j)} \in V$ ($j \in [0; n_i{+}1]$), $v_{\pi(0)}$ and $v_{\pi(n_i+1)}$ =0 (representing the depot) and n_i is the number of customers of route R_i. The total cost of a solution, which is the objective measure $f()$, is calculated as Formula (2).

$$f = \sum_{i=1}^{m} C(R_i) \tag{2}$$

where the total demand of each route should not exceed the vehicle capacity

$$Demand(R_i) = \sum_{j=1}^{n_i} q_\pi(j) \leq Q_i \tag{3}$$

where $q_\pi(j)$ is the quantity requested by customer $\pi(j)$, and Q_i is the capacity of vehicle i. Beside the capacity constraint, a route is only feasible if the total travelling time does not exceed the given global upper bound T:

$$Time(R_i) = \sum_{j=0}^{n_i} c_{\pi(j),\pi(j+1)} + \sum_{j=1}^{n_i} \delta_{\pi(i)} \leq T \tag{4}$$

2.2. Dynamic Vehicle Routing Problems (DVRP)

In real-life situations, the travel time between customers depends on traffic condition of the current road network. This means we do not have a predetermined speed set for all vehicles. Traffic could vary significantly depending on the time of the day. For example, the travel time during rush hour would be multiple times higher than the time travelling at midnight. This work uses the DVRP model proposed in [26], [21]. In this model, a traffic factor (t_{ij}) is added to all edges connecting customers v_i and v_j. The cost between customers v_i and v_j is now assigned as follows:

$$c_{ij} = d_{ij} \times t_{ij} \tag{5}$$

where the traffic factor t_{ij} is generated using the formula below:

$$t_{ij} = \begin{cases} t_{ij} \longleftarrow 1 + Rnd \in [F_L, F_U] & \text{if } rr \leq mt \\ t_{ij} \longleftarrow 1 & \text{otherwise} \end{cases}$$

where

- mt is the magnitude of changes which represents the probability of adding a traffic factor value to current edge (route).
- F_L is the lower bound value of the traffic factor.
- F_U is the upper bound value of the traffic factor.
- rr is uniform random number between zero and one, $rr \in [0, 1]$.

The model generates a random number *Rnd* for every f time intervals. The value of *Rnd* is between F_L and F_U. Each edge will be assigned with a traffic factor based on the value of the magnitude of change (mt). The traffic factor value could be t_{ij}=1+ *Rnd* or t_{ij}=1. So *Rnd* is the congestion level. The higher value the more congestion there will be. The value of *Rnd* could be close or equal to F_U, simulating rush hour traffic. During off peak hour *Rnd* should be close or equal to F_L. While *Rnd* affects the severity of congestion, the parameter f affects the frequency of congestion. Due to this dynamic change, the travel time between customers is asymmetric. That means the travel time from v_i to v_j might not the same of that from v_j to v_i. Hence it is highly desirable to have an effective optimisation algorithm for DVRP that can accommodate these changes and quickly find high quality solutions for new situations.

Existing solutions for DVRP can be categorised into two types: re-optimisation methods and adapting old solution methods. The re-optimisation methods consider the arrival of new information (a new traffic) as a new problem that needs to be solved from scratch. However, re-optimisation methods impractical for real world applications because solving problem from scratch is time consuming, the solution to the new problem should not be very different, and a very small change should not be considered as a new problem. Adapting old solution methods, on the other hand, attempt to reduce the computational complexity of DVRP by re-using the knowledge from the past to accommodate new changes into existing solution, thus much more practical solutions to DVRP. That is, once a change occurs, the method has already generated a number of solutions that can be used to reduce the computational complexity and help to integrate new changes. Since the arrival of a new traffic in DVRP can not be considered as a totally new DVRP, the work proposed in this paper adapts old solution to accommodate the arrival of new changes.

3. Methodology

Population based algorithm is a good choice for dynamic problems like DVRP. It is often referred as evolutionary algorithms (EAs). EAs such as genetic algorithm [16], [19], particle swarm optimisation [14], [34], [47] and differential evolution [35], [48], [8], have been successful in a wide range of difficult real world problems [7]. As mentioned early one challenge of EAs is the adaptive configuration which allows the search process to be tailored according to its current state [36]. EAs often work well only for the problem

they have been configured and customised for [18]. It is difficult for a fixed configuration to always be the best during the whole search process. More importantly best configurations do change when changes occur in the problem and its constraints.

One way to address the above issue is self-adaptive methods, which have appeared in [41]. The existing self-adaptive methods focus on evolving one or a few operators or parameter values while our proposed method for DVRP considers extra variations including the combination of operators and the sequence of applying operators. The proposed self-adaptive framework is integrated with genetic algorithm (GA). It searches for the best GA configurations in order to obtain better DVRP solutions. To improve the convergence, a range of improvement operators are introduced, which are also evolved along with the GA configuration. This proposed self-adaptive EA will not only reduce the need for manual parameter tuning but also allows the configuration to be dynamically and automatically adjusted to suit the dynamic changes and lead the search toward promising regions in the solution space.

Before we describe the details of the proposed self-adaptive EA, the GA components for DVRP are briefly explained.

3.1. Genetic Algorithm Component

Genetic algorithm (GA) is a well-known stochastic optimisation algorithm [13]. It operates on a population of solutions and each solution is encoded in a chromosome. These solutions are iteratively improved through the application of operators including crossover and mutation. The flowchart of GA is shown in Figure 1. The major steps are also described.

- **1 Set parameters**. The main parameters of GA are initialised in this step. They are:
  - Maximum number of generations ($MaxGen$).
  - Population size (PS). This is the number of individuals in the population.
  - Crossover rate (CR). This parameter determines the probability of applying the crossover operator on individuals.
  - Mutation rate (MR). This sets the probability of mutating the current solution.

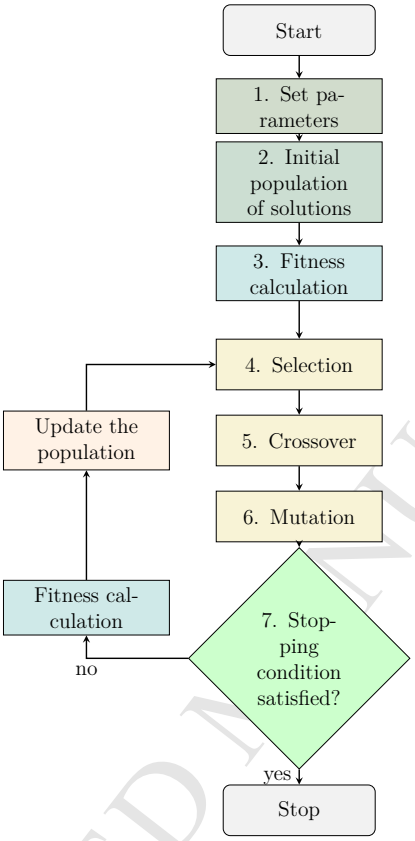


Figure 1: The flowchart of GA

- **2 Solution representation and population initialisation**. To handle DVRP by GA, each solution is represented as a one-dimensional array, the chromosome as shown in Figure 2. The value of each cell represents a customer C_i. For example, in Figure 2 we have three routes denoted as: R_1, R_2 and R_3. R_1 visits four customers, C_2, C_4, C_1 and C_5. R_2 has three customers, C_7, C_9 and C_{11} on its path. R_3 is set to serve four customers, C_6, C_8, C_3 and C_{10}. All customers are visited following their order in the given route. In this work, the population of solutions, with size equal to PS, is randomly initialised.

- **3 Fitness evaluation**. In this part each solution in the population will be evaluated against the problem instance to generate the fitness value. These values direct the search towards better solutions. In

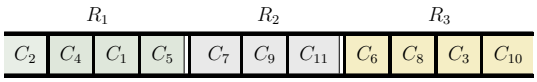


Figure 2: Solution representation for DVRP

DVRP, the fitness is simply the total travelling time on visiting all customers calculated using Equation (2).

- **4 Selection**. The selection mechanism picks a proportion of solutions from existing population for the reproduction or crossover process. Fitter solutions have higher chances to be selected.

- **5 Crossover**. Crossover is the main operator in the reproduction step. It creates a new population of solutions by combining the genetic materials of the selected solutions.

- **6 Mutation**. Mutation is another key operator of reproduction. It helps increase the diversity so the search process may escape from a local optima. Mutation in our method modifies solutions generated by crossover although it could be independent from crossover.

- **7 Stopping condition**. This checks whether the search should continue. If one of the stopping conditions is met, for example the maximum generation $MaxG$ has been reached, then the GA search process will stop and the best solution found will be returned as the final solution. Otherwise, the GA process will continue with the following steps:

  - Fitness calculation: calculate the fitness value of the new generated solutions.
  - Update the population: replace the old solutions by new ones if they are better in term of the fitness values.
  - Go to *Step 4*.

3.2. Self-adaptive EA

The proposed Self-adaptive EA (denoted as SAEA) aims to dynamically adjust GA configurations including operators and parameters to suit the DVRP. Please note that we will use the term "*individual*" within the proposed algorithm in order to differentiate it from the actual DVRP solution. Thus,

each SAEA individual consists of DVRP solution and its configuration. The proposed SAEA takes the advantage of the complementaries and complex interactions between various configurations. By allowing EA configuration to be evolved along with the population of individuals, the proposed algorithm can effectively deal with DVRP through generating different configuration for different landscape changes and thus guide the search process towards promising area to attain high quality solution. The flowchart of the proposed SAEA is shown in Figure 3. Comparing with Figure 1, SAEA is based on classical GA with some key components which are highlighted in Figure 3. After selection SAEA calls a decoder (Section 3.2.5) to turn the selected individuals into DVRP solution and configuration. The configuration will be applied to the solution (Section 3.2.6). After this step, the procedure is the same as GA which is checking stopping condition, evaluating solution, updating the population and then back to selection for the next generation.

The details of the proposed self-adaptive EA framework are described below.

3.2.1. Solution representation

Each individual in SAEA is also represented as a one-dimensional array which has three parts as shown in Figure 4.

Part 1 - Solution: This part represents a DVRP solution. We use path representation with route delimiters. A sequence of customer indices with delimiters represents the beginning and the end of route. For example, this Part 1 sequence $|D|C1|C3|C4|D|C2|C6|C5|D|$ means there are two routes. The first one serves customers $C1$, $C3$, and $C4$ and they are visited in the order of the given sequence. The second route consists of customers $C2$, $C6$, and $C5$. It starts from $C2$, and ends at $C5$. D is the deport which is also the delimiter to separate routes. During the search process, the proposed SAEA uses the given DVRP solution along with evolved configuration to generate a new DVRP solution.

Part 2 - Parameters: This part encodes numeric parameters. In this study two different parameters are used which are P_1 and P_2, the crossover rate (CR) and the mutation rate (MR), respectively. The possible values are provided in this study as $P_1 \in \{0.2, 0.4, 0.6, 0.8\}$ and $P_2 \in \{0.3, 0.5, 0.7, 0.9\}$. During the optimisation process, a parameter will be assigned with one value from the given set. Note that mutation and crossover are independent in our approach, so there is no constraint that mutation rate plus crossover rate should be 100%. During the optimisation process, the proposed SAEA will

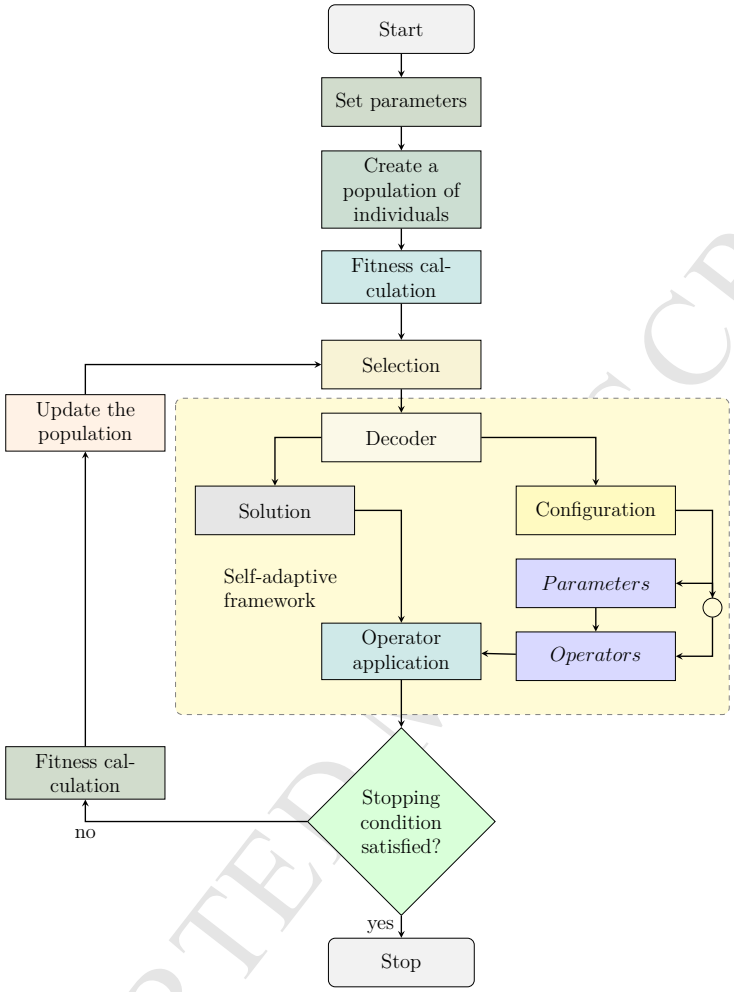


Figure 3: The flowchart of the proposed Self-adaptive EA framework

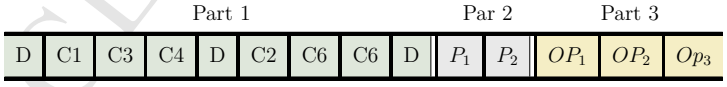


Figure 4: Solution representation

assign, for each parameter, one value from the given set.

Part 3 - Operators: This part encodes the operators which include three types of operators which are denoted as OP_1, OP_2 and OP_3.

OP_1. This represents different type of crossover operators, which exchange genetic materials of the selected solutions to create offspring's. The solutions undergo the crossover operator based on the CR value. In this work, OP_1 uses three different types of crossover operators as follows [32]:

- OP_1_1 Order-based operator: This operator randomly sets two crossover points on one selected solution, copies the part in between the two points into the offspring and then fill in the remaining parts using the corresponding elements from the second selected solution.
- OP_1_2 Route-based operator: This operator selects the best k (e.g., k=2) routes in term of fitness value from both selected solutions, moves the selected routes into the offspring. The remaining positions are filled by following the customer sequence on the best solution.
- OP_1_3 Swap-based operator: This operator also randomly select k (e.g., k=2) routes from both solutions. However it swaps the selected routes between two solutions and remove duplicate customers and re-insert missed ones into the best possible position.

It should be noted that the above crossover operators ensure the feasibility by removing duplicated customers and inserting missed ones.

OP_2. This operator is responsible for mutation. It moves one or few positions in the given offspring to new locations based on MR value. In this work, OP_2 make use of the following three mutation operators [32]:

- OP_2_1 Random remove: This mutation operator removes N customers (e.g., N=3) at random and re-assigns them to different positions.
- OP_2_2 Worst remove: This one removes N customers with the largest cost savings and re-assigns them to best cost saving positions.
- OP_2_3Reverse: It selects one route at random and then shuffles customer orders.

OP_3. This operator iteratively improves on the DVRP solution acting like a local search operator. A given solution is modified by a neighbourhood operator, seeking for better alternatives which may have better fitness values. In this work, the following operators are incorporated in SAEA. They have appeared in early VRP works in [2]:

- OP_3_1 Swap: It iteratively selects two customers from two different routes, and swap them. If the swap leads to a better fitness value, then the search will continue based on the new solution generated from the swap.
- OP_3_2 Single move: It iteratively selects one customer at random and moves it to a different route. Only better solution will be accepted.
- OP_3_3 Double move: This operator iteratively selects two customers at random and move them to a different route. Only solutions bringing improvement are accepted.

Operators (OP_3_1, OP_3_2 and OP_3_3) will stop if there is no improvement after ten consecutive iterations (see Section 4.2).

During the search process, the proposed algorithm evolves the combination of these operators, one operator of each type and their application sequence.

3.2.2. Population of Individuals

A population of individuals is initialised, of which the size is PS. Each individual has three parts, Part 1, Part 2 and Part 3, as shown in Figure 4. These parts are initialised as follows:

Initialise Part 1 ($DVRPsolutions$). DVRP solutions are randomly created. We first create an empty route. Un-routed customers are randomly selected and added to the current route as long as there is no violation in the vehicle capacity. If no customer can be added into the current route, a new route will be created. The process will be repeated until all customers are routed.

Initialise Part 2 (P_1 and P_2). Both P_1 and P_2 are initialised by randomly assigning a value from the predefined sets so $P_1 = Random\{0.2, 0.4, 0.6, 0.8\}$ and $P_2 = Random\{0.3, 0.5, 0.7, 0.9\}$.

Initialise Part 3 (OP_1, OP_2 and OP_3). Each operator, OP_1, OP_2 or OP_3, is randomly assigned one operator from the corresponding operator type. OP_1=$Random${set of crossover operators}, OP_2=$Random${set of mutation operators}, and OP_3=$Random${set of improvement operators}.

3.2.3. Fitness calculation

The fitness measure of each individual in DVRP represents the total travelling time of visiting all customers, as shown in Equation (2).

3.2.4. Selection

SAEA uses roulette wheel selection, which is a stochastic mechanism favouring good solutions in the population. The probability (P) of i^{th} individual to be selected into the mating pool is proportional to its fitness value (f), which is calculated as follows:

$$P(i) = \frac{f(i)}{\sum_{j=1}^{PS} f(j)} \tag{6}$$

where PS is the population size.

3.2.5. Decoder

The decoder splits a selected individual and turns its Part 1 as a DVRP solution, its Part 2 as parameters and its Part 3 as the operators. The main role each part is explained in Section 3.2.1.

3.2.6. Operator application

The operator application performs the following tasks on the decoded individual from the last step:

- **Task 1**. This task is performed on Part 1 (DVRP solution) of the selected individuals. It takes the selected DVRP solutions as input and then applies the evolved sequence of operators (crossover, mutation and local search) to the solution to generate a new offspring.
- **Task 2**. This task is performed on Part 2 (GA parameters, P_1 and P_2). It uses the mutation operator to generate a new values for P_1 and P_2.
- **Task 3**. This task is performed on Part 3 (GA variation operators (crossover and mutation) and improvement operator, OP_1 OP_2 and OP_3). It applies the crossover and mutation operators to change the sequence of all operators or to generate a new ones. To generate a new sequence of operators, we apply the single point crossover operator. We first randomly select a crossover point and then the string from beginning up to the crossover point is copied from the first individual and the rest is copied from the second individual. The mutation operator randomly selects one operator from the corresponding operator type as follows: OP_1=*Random*{set of crossover operators}, OP_2=*Random*{set of mutation operators}, and OP_3=*Random*{set of improvement operators}.

The rest of SAEA process is identical to that in classical GA discussed in the last sub-section.

4. Experiments

This section firstly presents the benchmark instances, then the parameter settings and lastly the performance measure.

4.1. Benchmark instances

We use two dynamic routing problems traffic congestion for evaluation: dynamic vehicle routing problem (DVRP) and dynamic travelling salesman problem (DTSP) [20].

4.1.1. DVRP instances

The DVRP benchmark instances are published by [26]. The instances are generated using three well-known static VRP instances. Table 1 shows the main features of these instances.

Table 1: Static VRP instances for generating DVRP

Instance	No. of customers	Capacity	No. of vehicles
F-n45-k4	45	2010	4
F-n72-k4	72	30000	4
F-n135-k7	135	2210	7

To simulate various congestion levels, different values for the frequency of changes (f) and the magnitude of changes (m) were used. That is, both f and m are randomly generated within the pre-defined range to simulate the traffic level on different time of the day. Similar to [26], we set these parameters as follows to generate DVRP instances: $f = Random(1, 100)$, $mt = Random(0, 1)$, $F_L = 0$ and $F_U = 5$. (See explanation of these parameters in Section 2)

4.1.2. DTSP instances

The travelling salesman problem (TSP) is a well-known routing problem as well. Given a set of customers and their positions (pairwise distances), the goal is to find the shortest path, where each customer is visited only once and the path ends at the first customer. The total travelling time

is the measure for performance. TSP with traffic congestion is a dynamic version of TSP (denoted as DTSP). DTSP benchmark is reported in [20]. The DTSP instances are take from the TSBLIB and converted into DTSP through randomly changing the travel time between customers. The total number of customers, or the size of instances is shown in Table 2. To simulate real world situation, the magnitude of changes (mt) is randomly selected from three different ranges that represent *small*, *medium* and *high* magnitude of environment changes; $m = Random(0, 0.25)$, $m = Random(0, 0.5)$ and $m = Random(0, 1)$. For each mt, two values for the frequency (f) of changes are used, f=15 seconds and f=30 seconds.

Table 2: TSP instances for generating DTSP

Instances	Size
Berlin52	52
Eill101	101
D198	198
Lin318	318
Pcb442	442
U574	574
Rat783	783
Pcb1173	1173

4.2. SAEA settings

The proposed SAEA has only three main parameters that need to be set: the maximum number of generations ($MaxGen$), population size (PS) and the number of customers to be removed (k). To ensure a fair comparison, $MaxGen$ is fixed to 1,000 fitness evaluations like the other algorithms in the literature [26]. PS and k were fine tuned by conducting experiments on benchmark instances for different sets of values, PS=30 and k=3.

4.3. Performance measure

To facilitate comparisons on DVRP, the number of runs on each instance and the performance measure are the same as that in [26]. We conducted 30 independent runs using different random seeds. The $offline$ performance is used in our measure. The calculation of this measure is from [26] and shown below.

$$P_{off} = \frac{1}{MaxGen} \sum_{i=1}^{MaxGen} \left(\frac{1}{R} \sum_{j=1}^{R} P_{ij} \right) \quad (7)$$

where $MaxGen$ is the total number of generations, R is the maximum number of runs, and P_{ij} is the quality of the best solution in the population after the change at i^{th} iteration of j^{th} run. As for DTSP, the performance measure is simply the average travel time of the best solutions from 30 independent runs from each environment [20].

5. Results and discussions

This section presents the experimental results and analysis of the proposed SAEA. It is divided into four subsections. In Section 5.1, we investigate the benefit of self-adaptive mechanism. The computational results of using different traffic update formula is presented in Section (5.2). Sections 5.3.1 and 5.3.2 show the comparisons between SAEA with the state of the art algorithms on both DVRP and DTSP. Further discussion on SAEA performance is presented in Section 5.4.

5.1. Effectiveness evaluation

To show the effectiveness of the self-adaptive mechanism in SAEA, we have tested SAEA using each operator separately, fixed parameter values and without any of these self-adaptive elements. Hence eleven variations of EA are introduced:

- $SAEA$: the proposed self-adaptive algorithm (see Section 3).
- EA_1: uses fixed parameter values (CR=0.85 and MR=0.03) and fixed operators (OP_1=OP_1_1 and OP_2=OP_2_1). We also excluded OP_3 from the configuration in order to have a clear indication on operator combination impact.
- EA_2: same as EA_1 but OP_1=OP_1_2 and OP_2=OP_2_2
- EA_3: same as EA_1 but OP_1=OP_1_3 and OP_2=OP_2_3
- EA_4: uses the three different operators, OP_1=OP_1_1, OP_2=OP_2_1 and OP_3= OP_3_1

- EA_5: same as EA_4 but OP_3=OP_3_2
- EA_6: same as EA_4 but OP_3=OP_3_3
- EA_7: uses fixed operators (randomly selected from the given set) but the parameter values are randomly changed.
- EA_8: same as EA_7 but MR= 0.3.
- EA_9: same as EA_7 but MR= 0.5.
- EA_{10}: same as EA_7 but MR= 0.7.
- EA_{11}: same as EA_7 but MR= 0.9.

Together with SAEA these eleven algorithms (EA_1 to EA_{11}) were tested on the same DVRP benchmark instances, under the same environment with the same settings. Table 3 shows the best, average and the standard deviation (Std) of these twelve algorithms from 30 independent runs. The best among the twelve are highlighted in bold. Clearly we can see that SAEA is better than all other alternative. The best, average and the standard deviation obtained by $SAEA$ are all the lowest among results from all algorithms. This comparison shows that the self-adaptive mechanism is effective.

To verify the observation from Table 3, we also did statistical significance test using Wilcoxon signed-rank test with 0.05 confidence level between SAEA and other eleven alternatives. The p-values of SAEA versus each one of them on different instances are presented in Table 4. A value less than 0.05 (p-value <0.05) means $SAEA$ is statistically better than its counterpart. A value greater than 0.05 (p-value >0.05) means the opposite. As we can see Table 4 has no value greater than 0.05 present. It shows that $SAEA$ is indeed significantly better than other variations. This again verifies the benefit of the proposed self-adaptive mechanism.

5.2. Results of different traffic update formula

In this section, the proposed algorithm ($SAEA$) and the seven algorithms (EA_1 to EA_7) were tested on the same DVRP benchmark instances, under the same environment with the same settings but using different traffic updates formula. In this formula, the traffic level is randomly either decreased or increased. The average and the standard deviation (Std) of these eight

algorithms over 30 independent runs are presented in Table 5. The best average objective function value and the standard deviation for each instance has been highlighted in boldfont. Table 5 shows that, compared with the other seven referenced algorithms, $SAEA$ produced better average and the standard deviation across all instances.

5.3. Compared to the state of the art algorithms

In the following subsection, the results of $SAEA$ for DVRP and DTSP are compared with the best reported results in the literature.

5.3.1. Comparison on DVRP

The existing approaches for DVRP used in this comparison are listed below. They all have been tested on same DVRP instances.

1. ASrank-CVRP: rank-based ant system[3], [26].
2. ACS-DVRP: standard ant colony system [27], [26].
3. M-ACO: memetic ant colony optimisation [25].
4. MMASR: $\mathcal{MAX} - \mathcal{MIN}$ ant system [23].
5. RIACO: random immigrants ant colony optimisation [21].
6. EIACO: elitism-based immigrants ant colony optimisation [21].
7. MIACO: memory-based immigrants ant colony optimisation [22].

The best value of $SAEA$ over 30 runs and that from other algorithms are presented and compared in Table 6. Value in Bold indicates the best result on the instance. From Table 6 we can see that $SAEA$ is better than results reported by other methods on all tested instances. It should be noted that our $SAEA$ and other algorithms use the same number of fitness evaluations as the search termination condition. This means that the total computational time for all algorithms is the same or very similar. For all instances, we are able to achieve results better than the current best. This suggests that $SAEA$ has the ability to cope with the dynamic changes and to find high quality solutions.

5.3.2. Comparison on DTSP

The DTSP algorithms from the literature that were used for our comparison are listed below. They all have tested on the same DTSP instances as $SAEA$.

1. MMAS: ant colony optimization with local search [20].

2. KP: local search algorithm [17], [20].
3. MM: $\mathcal{MAX} - \mathcal{MIN}$ ant system and local search [42], [20].
4. P-ACO: population based ant colony optimization [12], [20].
5. EIACO: ant colony optimization with immigrants schemes [24].
6. GAPX: elitism-based immigrants genetic algorithm[43], [20].

$SAEA$ and the state-of-the-art algorithms all terminate after 10 environmental changes. All methods conducted 30 independent runs. The average of $SAEA$ results are presented in Table 7 and compared with that from MMAS, KP, MM, P-ACO, EIACO and GAPX. The best result on one instance, e.g. the best in its row, is highlighted in bold. As Table 7 shows, $SAEA$ achieved new best results on 41 out of 48 cases. From Table 7, we infer that, although $SAEA$ does not obtain the best average results for 7 instances, overall, the average results of these instances with regard to relative error are relatively small. Considering the magnitude of changes (mt) and the frequency of changes (f), we can make the following observation:

- When f change every 15 seconds, $SAEA$ is better than the compared algorithms (MMAS, KP, MM, P-ACO, EIACO and GAPX) on all instances for $m \in [0, 0.25]$ and $m \in [0, 1]$. As for $m \in [0, 0.5]$, $SAEA$ obtained better results on 3 instances and being inferior on 5 instances. The most competing algorithms are MMAS and KP. Considering an individual comparisons for f=15 seconds $SAEA$ is better than MMAS and KP on 18 and 23 out of 24 tested instances, respectively.

- When f change every 30 seconds, $SAEA$ outperforms the compared algorithms (MMAS, KP, MM, P-ACO, EIACO and GAPX) on all instances for $m \in [0, 0.25]$, $m \in [0, 0.5]$, and $m \in [0, 1]$, with the exception of one instance (*Berlin52* when $m \in [0, 0.5]$).

The above results show that $SAEA$ outperforms other DTSP algorithms on most instances. For other instances it is still very competitive. Also, it is worth noting that the existing methods are specifically designed to produce the best results only for one or a few instances, while our SAEA incorporates no domain specific knowledge or preference and generalize well over all instances.

5.4. Discussion

The results presented throughout this work illustrate that, across two different combinatorial optimisation problems, *SAEA* produced very good results compared to the state of the art algorithms. More importantly, *SAEA* obtained new best results for all DVRP instances and 85.4% for DTSP instances. We hypothesize that the capability of *SAEA* in dealing with two different dynamic routing problems and achieving good results is due to the following two factors.

- The ability of the proposed algorithm in handling the dynamic changes by evolving different configuration to be applied at different decision point. By evolving configuration, the search can effectively capture different area in the search and thus problem changes can be accommodated and tracked.

- The ability of the proposed algorithm in generating, for each solution, different sequences of operator combination during the search. By doing so, the proposed algorithm can effectively escape from the local optima and explore different areas in the problem solution search space.

6. Conclusion

In this paper, we presented a self-adaptive evolutionary algorithm for the the dynamic vehicle routing problem with traffic congestion. The proposed algorithm adaptively evolves the configurations (the parameter values and operator types) of evolutionary algorithm in order to effectively handle the dynamic changes as well as generating high quality solutions. It encodes various configurations that undergo a thorough evolutionary process and evolve at the same time than the solution to a given problem. The proposed algorithm has been tested on two routing problems with traffic congestion: vehicle routing and travelling salesman. The obtained results demonstrate the effectiveness of the proposed algorithm when compared to the one without self-adaptive mechanisms. It also achieved competitive, if not better, results compared to other algorithms proposed in the literature. In our future work, we would like to apply the proposed algorithm on a real-world dynamic vehicle routing problem.

[1] Baker, B. M. and Ayechew, M. (2003). A genetic algorithm for the vehicle routing problem. *Computers & Operations Research*, 30(5):787–800.

[2] Bräysy, O. and Gendreau, M. (2005). Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation science*, 39(1):104–118.

[3] Bullnheimer, B., Hartl, R. F., and Strauss, C. (1999). Applying the ant system to the vehicle routing problem. In *Meta-Heuristics*, pages 285–296. Springer.

[4] Dang-Nhu, R., Dardinier, T., Doerr, B., Izacard, G., and Nogneng, D. (2018). A new analysis method for evolutionary optimization of dynamic and noisy objective functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1467–1474. ACM.

[5] Dantzig, G. B. and Ramser, J. H. (1959). The truck dispatching problem. *Management science*, 6(1):80–91.

[6] Eiben, A. E. and Smit, S. K. (2011). Parameter tuning for configuring and analyzing evolutionary algorithms. *Swarm and Evolutionary Computation*, 1(1):19–31.

[7] Eiben, A. E. and Smith, J. (2015). What is an evolutionary algorithm? In *Introduction to Evolutionary Computing*, pages 25–48. Springer.

[8] Fang, H., Zhou, A., and Zhang, H. (2018). Information fusion in offspring generation: A case study in de and eda. *Swarm and Evolutionary Computation*.

[9] Fialho, Á., Da Costa, L., Schoenauer, M., and Sebag, M. (2010). Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence*, 60(1-2):25–64.

[10] Gambardella, L. M., Taillard, É., and Agazzi, G. (1999). Macs-vrptw: A multiple colony system for vehicle routing problems with time windows. In *New ideas in optimization*. Citeseer.

[11] Gendreau, M., Hertz, A., and Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management science*, 40(10):1276–1290.

[12] Guntsch, M. and Middendorf, M. (2002). Applying population based aco to dynamic optimization problems. In *International Workshop on Ant Algorithms*, pages 111–122. Springer.

[13] Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. U Michigan Press.

[14] Hu, W. and Tan, Y. (2016). Prototype generation using multiobjective particle swarm optimization for nearest neighbor classification. *IEEE Transactions on Cybernetics*, 46(12):2719–2731.

[15] Imran, A., Salhi, S., and Wassan, N. A. (2009). A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem. *European Journal of Operational Research*, 197(2):509–518.

[16] Ji, B., Yuan, X., and Yuan, Y. (2017). Modified nsga-ii for solving continuous berth allocation problem: Using multiobjective constraint-handling strategy. *IEEE Transactions on Cybernetics*, PP(99):1–11.

[17] Kanellakis, P.-C. and Papadimitriou, C. H. (1980). Local search for the asymmetric traveling salesman problem. *Operations Research*, 28(5):1086–1099.

[18] Karafotias, G., Hoogendoorn, M., and Eiben, A. E. (2015). Parameter control in evolutionary algorithms: Trends and challenges. *IEEE Trans. Evolutionary Computation*, 19(2):167–187.

[19] Martarelli, N. J. and Nagano, M. S. (2018). A constructive evolutionary approach for feature selection in unsupervised learning. *Swarm and Evolutionary Computation*.

[20] Mavrovouniotis, M., Muller, F. M., and Yang, S. (2016). Ant colony optimization with local search for dynamic traveling salesman problems. *IEEE Transactions on Cybernetics*, PP(99):1–14.

[21] Mavrovouniotis, M. and Yang, S. (2012a). Ant colony optimization with immigrants schemes for the dynamic vehicle routing problem. In *European Conference on the Applications of Evolutionary Computation*, pages 519–528. Springer.

[22] Mavrovouniotis, M. and Yang, S. (2012b). Ant colony optimization with memory-based immigrants for the dynamic vehicle routing problem. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE.

[23] Mavrovouniotis, M. and Yang, S. (2013a). Adapting the pheromone evaporation rate in dynamic routing problems. In *European Conference on the Applications of Evolutionary Computation*, pages 606–615. Springer.

[24] Mavrovouniotis, M. and Yang, S. (2013b). Ant colony optimization with immigrants schemes for the dynamic travelling salesman problem with traffic factors. *Applied Soft Computing*, 13(10):4023–4037.

[25] Mavrovouniotis, M. and Yang, S. (2013c). Dynamic vehicle routing: A memetic ant colony optimization approach. In *Automated Scheduling and Planning*, pages 283–301. Springer.

[26] Mavrovouniotis, M. and Yang, S. (2015). Ant algorithms with immigrants schemes for the dynamic vehicle routing problem. *Information Sciences*, 294:456–477.

[27] Montemanni, R., Gambardella, L. M., Rizzoli, A. E., and Donati, A. V. (2005). Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization*, 10(4):327–343.

[28] Neri, F. and Cotta, C. (2012). Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14.

[29] Neumann, F. and Witt, C. (2015). On the runtime of randomized local search and simple evolutionary algorithms for dynamic makespan scheduling. In *IJCAI*, pages 3742–3748.

[30] Osman, I. H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of operations research*, 41(4):421–451.

[31] Pillac, V., Gendreau, M., Guéret, C., and Medaglia, A. L. (2013). A review of dynamic vehicle routing problems. *European Journal of Operational Research*, 225(1):1–11.

[32] Potvin, J.-Y. (2009). State-of-the art review-evolutionary algorithms for vehicle routing. *INFORMS Journal on Computing*, 21(4):518–548.

[33] Pourhassan, M., Gao, W., and Neumann, F. (2015). Maintaining 2-approximations for the dynamic vertex cover problem using evolutionary

algorithms. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 903–910. ACM.

[34] Rubio-Largo, Á., Vanneschi, L., Castelli, M., and Vega-Rodríguez, M. A. (2018). Swarm intelligence for optimizing the parameters of multiple sequence aligners. *Swarm and Evolutionary Computation*.

[35] Sabar, N. R., Abawajy, J., and Yearwood, J. (2017a). Heterogeneous cooperative co-evolution memetic differential evolution algorithm for big data optimization problems. *IEEE Transactions on Evolutionary Computation*, 21(2):315–327.

[36] Sabar, N. R., Ayob, M., Kendall, G., and Qu, R. (2015). Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems. *IEEE Transactions on Evolutionary Computation*, 19(3):309–325.

[37] Sabar, N. R., Chung, E., Tsubota, T., de Almeida, P. E. M., et al. (2017b). A memetic algorithm for real world multi-intersection traffic signal optimisation problems. *Engineering Applications of Artificial Intelligence*, 63:45–53.

[38] Sabar, N. R., Song, A., and Zhang, M. (2016a). A variable local search based memetic algorithm for the load balancing problem in cloud computing. In *European Conference on the Applications of Evolutionary Computation*, pages 267–282. Springer.

[39] Sabar, N. R., Turky, A., and Song, A. (2016b). A multi-memory multi-population memetic algorithm for dynamic shortest path routing in mobile ad-hoc networks. In *Pacific Rim International Conference on Artificial Intelligence*, pages 406–418. Springer.

[40] Serpell, M. and Smith, J. E. (2010). Self-adaptation of mutation operator and probability for permutation representations in genetic algorithms. *Evolutionary Computation*, 18(3):491–514.

[41] Smith, J. E. (2012). Self-adaptative and coevolving memetic algorithms. In *Handbook of Memetic Algorithms*, pages 167–188. Springer.

[42] Stutzle, T. and Hoos, H. (1997). Max-min ant system and local search for the traveling salesman problem. In *Evolutionary Computation, 1997., IEEE International Conference on*, pages 309–314. IEEE.

[43] Tinós, R., Whitley, D., and Howe, A. (2014). Use of explicit memory in the dynamic traveling salesman problem. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 999–1006. ACM.

[44] Toth, P. and Vigo, D. (2014). *Vehicle Routing: Problems, Methods, and Applications*, volume 18. SIAM.

[45] Turky, A. M., Abdullah, S., and Sabar, N. R. (2014). A hybrid harmony search algorithm for solving dynamic optimisation problems. *Procedia Computer Science*, 29:1926–1936.

[46] Vidal, T., Crainic, T. G., Gendreau, M., and Prins, C. (2013). Heuristics for multi-attribute vehicle routing problems: a survey and synthesis. *European Journal of Operational Research*, 231(1):1–21.

[47] Zhong, Y., Lin, J., Wang, L., and Zhang, H. (2018). Discrete comprehensive learning particle swarm optimization algorithm with metropolis acceptance criterion for traveling salesman problem. *Swarm and Evolutionary Computation*.

[48] Zhou, Y. Z., Yi, W. C., Gao, L., and Li, X. Y. (2017). Adaptive differential evolution with sorting crossover rate for continuous optimization problems. *IEEE Transactions on Cybernetics*, PP(99):1–12.

Table 3: SAEA Comparing with 11 Variations

Algorithm	Instance	F-n45-k4	F-n72-k4	F-n135-k7
$SAEA$	Best	**1097.24**	**418.76**	**2411.53**
	Average	**1101.833**	**426.61**	**2425.55**
	Std	**3.68**	**5.46**	**12.94**
EA_1	Best	1127.52	491.34	2584.54
	Average	1135.13	511.52	2647.31
	Std	11.42	27.92	26.44
EA_2	Best	1206.35	476.58	2611.23
	Average	1327.28	508.36	2753.92
	Std	25.31	22.83	23.46
EA_3	Best	1213.44	481.75	2592.53
	Average	1347.35	510.42	2686.03
	Std	27.41	32.62	27.49
EA_4	Best	1196.35	483.09	2581.54
	Average	1281.78	507.01	2622.4
	Std	22.93	19.75	31.27
EA_5	Best	1112.62	462.57	2542.47
	Average	1273.56	495.64	2679.18
	Std	22.74	16.42	36.21
EA_6	Best	1188.27	486.56	2541.65
	Average	1291.35	514.74	2667.46
	Std	25.43	26.11	33.08
EA_7	Best	1231.46	492.57	2541.26
	Average	1368.15	548.96	2668.11
	Std	28.01	20.72	34.53
EA_8	Best	1245.22	501.27	2576.10
	Average	1332.02	563.51	2604.95
	Std	25.28	22.04	31.23
EA_9	Best	1305.37	530.47	2712.47
	Average	1412.15	606.62	2927.48
	Std	33.70	31.06	40.64
EA_{10}	Best	1353.24	590.27	2703.63
	Average	1472.04	640.08	2922.84
	Std	32.07	36.92	41.67
EA_{11}	Best	1373.24	557.43	2723.07
	Average	1475.46	607.02	2897.63
	Std	34.02	38.16	40.62

Table 4: The p-value of SAEA compared to other algorithms

$SAEA$ $vs.$	EA_1	EA_2	EA_3	EA_4	EA_5	EA_6	EA_7	EA_8	EA_9	EA_{10}	EA_{11}
Instance	p-value	p-value	p-value	p-value	p-value	p-value	p-value	p-value	p-value	p-value	p-value
F-n45-k4	0.000	0.010	0.041	0.011	0.000	0.007	0.041	0.000	0.020	0.000	0.011
F-n72-k4	0.000	0.000	0.021	0.001	0.005	0.043	0.028	0.000	0.000	0.000	0.000
F-n135-k7	0.000	0.032	0.011	0.000	0.000	0.000	0.000	0.011	0.000	0.010	0.000

Table 5: The results of SAEA and other variants using different traffic update formula

Algorithm	Instance	F-n45-k4	F-n72-k4	F-n135-k7
SAEA	Best	**992.31**	**394.11**	**2285.11**
	Ava	**1141.24**	**421.23**	**2413.17**
	Std	**4.12**	**6.16**	**14.17**
EA1	Best	1048.27	427.16	2518.36
	Ava	1242.35	550.74	2875.14
	Std	16.55	32.74	30.19
EA2	Best	1114.63	468.39	2696.44
	Ava	1411.46	512.92	2974.56
	Std	29.11	28.25	27.27
EA3	Best	1196.34	469.56	2611.48
	Ava	1411.48	548.71	2811.39
	Std	30.12	31.1	29.2
EA4	Best	1114.21	481.27	2703.35
	Ava	1288.03	529.63	2901.28
	Std	23.24	18.11	34.09
EA5	Best	1137.62	471.68	2489.13
	Ava	1297.29	506.11	2718.26
	Std	23.12	17.1	37.08
EA6	Best	1100.79	461.56	2384.17
	Ava	1219.47	508.59	2667.46
	Std	21.11	20.92	34.29
EA7	Best	1104.46	461.44	2476.59
	Ava	1213.28	511.07	2674.25
	Std	25.17	18.94	31.31

Table 6: Comparing with State-of-the-art Methods on DVRP

Algorithm	F-n45-k4	F-n72-k4	F-n135-k7
$SAEA_{best}$	**1097.24**	**418.76**	**2411.53**
$SAEA_{Std}$	3.68	5.46	12.94
ASrank-CVRP	1126.69	473.27	2695.42
ACS-DVRP	1118.71	469.58	2656.18
M-ACO	1177.52	466.98	2618.42
MMASR	1112.49	461.09	2626.73
RIACO	1106.19	436.11	2496.29
EIACO	1117.49	429.21	2483.48
MIACO	1120.38	433.46	2505.39

Table 7: The computational results of $SAEA$ compared to other algorithms

Instances	$SAEA_{best}$	$SAEA_{Std}$	MMAS	KP	MM	P-ACO	EIACO	GAPX
m randomly generated $\in$ [0, 0.25] with f = 15 seconds								
Berlin52	7895.3	3.81	7895.3	**7865.3**	7923.2	8055.6	7916	8347.5
Eill101	**640.3**	14.8	652.5	657.2	655.7	671.4	663.2	703.4
D198	**16673.1**	11.7	16680.4	16690.1	16932	17679.6	17106.9	18203.2
Lin318	**44765.2**	15.7	44787	46019.9	45182.1	48588.1	46426.1	52737.9
Pcb442	**53682.7**	14.0	53741.3	55100.1	57226.9	60649	57460.5	65169.2
U574	**40211.4**	12.4	40255.7	40570.7	44604.1	47231.7	43510.8	47487
Rat783	**9457.3**	11.7	9534.6	9727.1	10996.3	11609	10543.9	11587.4
Pcb1173	**64344.7**	13.9	64412.2	65764.8	77971.7	81969.6	72742	84265.1
m randomly generated $\in$ [0, 0.5] with f = 15 seconds								
Berlin52	8292.4	12.5	**8285.8**	8259.3	8313.2	8506.4	8327.6	8715.8
Eill101	**692.6**	13.4	697.8	707.7	701.5	724.8	716.1	783.7
D198	17602.3	11.6	**17594.9**	17658	17783.6	18707.5	18134.6	19588.6
Lin318	47396.1	14.2	**47383.9**	48569.2	47779.3	51670.3	48981.4	56478.4
Pcb442	59345.6	15.4	**59320**	60088	63858.8	67403.6	63569.9	72879.2
U574	42992.9	15.2	**42976.2**	42969.9	47621.3	50416.9	46122.4	51026.5
Rat783	**10176.3**	12.3	10187.8	10311.2	11853.5	12447.3	11226.7	12826.6
Pcb1173	**73082.4**	10.8	73097.2	73564.4	88305.7	91737.9	81363.8	106143.2
m randomly generated $\in$ [0, 1] with f = 15 seconds								
Berlin52	**8585.1**	7.4	8594.9	8632.8	8603.8	8856.4	8634.5	9144.2
Eill101	**746.5**	8.2	750.8	770.4	755.2	784.8	768.1	843.2
D198	**18577.3**	8.1	18687.7	18591.4	18727.1	19774.5	19086.8	21127.5
Lin318	**49834.7**	8.2	49843.6	51711.4	50976.1	54641.3	51141.6	61448.2
Pcb442	**66062.8**	5.3	66066.1	66919.3	71487.4	75552.8	70377.6	80807.4
U574	**48469.1**	7.9	48487.9	48409.2	53248.6	56169.7	51682.8	58439.9
Rat783	**11956.3**	7.7	11971.9	12103.3	13671	14303.1	12948.5	15957.5
Pcb1173	**93664.2**	7.5	93686.7	93523.8	110622.2	114821.3	102621.4	145865
m randomly generated$\in$[0, 0.25] with f = 30 seconds								
Berlin52	**7859.7**	10.3	7897.3	7863.7	7911.4	8046.5	7919.7	8307.8
Eill101	**650.2**	3.4	650.7	655.6	653	669	661.5	696.7
D198	**16614**	7.6	16629.5	16651.6	16815.4	17477	17012.6	17855
Lin318	**44378.4**	12.3	44396.8	45905.8	44911	47616.6	46000.4	52337.2
Pcb442	**53736.2**	8.1	53754	54931	56211.6	58745.6	56936.4	64532.4
U574	**40048.1**	11.5	40062.6	40464.1	43030.7	46386.8	43033.9	46946.1
Rat783	**9465.9**	5.2	9487.3	9695.4	10748.8	11370.1	10416.3	11453.9
Pcb1173	**64244.5**	8.1	64259.4	65489.3	76034.4	80172.1	71838.2	81796.5
m randomly generated $\in$ [0, 0.5] with f = 30 seconds								
Berlin52	8254.1	3.5	8265.2	**8237.3**	8299.2	8473.4	8298.5	8672.1
Eill101	**691.2**	2.7	696.9	705.8	699	722.6	713.9	770.1
D198	**17564.7**	5.8	17572.5	17590.6	17651.1	18586.6	18071.1	19414.7
Lin318	**46914**	9.4	46923.8	48439.6	46993.1	50694	48300.3	56334.9
Pcb442	**59143.5**	10.3	59151	59959.6	62465.5	65975.1	62877.6	72674.3
U574	**42782.6**	7.4	42788.1	42816.2	46319.1	49472.7	45630.3	50736.3
Rat783	**10142.3**	6.1	10167.6	10297.3	11579.9	12206.9	11080.2	12426.6
Pcb1173	**72749.1**	7.8	72766.4	73327.9	86079.7	90127.3	80246.8	99820.1
m randomly generated $\in$ [0, 1] with f = 30 seconds								
Berlin52	**8573.3**	3.4	8584.2	8610.2	8587.3	8792.3	8627.4	8966.6
Eill101	**732.6**	1.9	749.1	768.2	752.6	819.2	765.8	830.5
D198	**18447.3**	6.2	18468.7	18494.3	18628.7	20196.5	19018.1	20625.8
Lin318	**48854**	7.7	48869.7	53866.6	49267.5	56111.2	50456	60855.7
Pcb442	**65822.6**	11.3	65828.3	66606.9	69335.8	76785	69731.3	80922.7
U574	**48427.8**	9.6	48445.2	48172.6	51791.3	56641.7	51119.9	58121.8
Rat783	**11911**	12.4	11914.3	11993	13341.6	14207.5	12776.5	15061.3
Pcb1173	**92892.4**	10.3	92916.7	93473.5	108006.6	112048.4	101139.6	136732.2