Chapter 15
Evolutionary Computation for Dynamic Capacitated Arc Routing Problem

Yi Mei, Ke Tang, and Xin Yao

Abstract. In this chapter, a new dynamic capacitated arc routing problem (CARP) is defined and investigated. Compared with the static CARP and other dynamic CARP investigated by the existing researches, the new dynamic CARP is more general and closer to reality, and thus is more worthwhile to be solved. Due to the stochastic factors included in the dynamic CARP, the objective is not to obtain the optimal solution in a specific environment, but to find a robust solution that shows good performance in all the possible environments. For the dynamic CARP, a robustness measure based on repair operator is defined. The corresponding repair operator is designed according to the real-world considerations. Then, the benchmark instances of the dynamic CARP are generated by extending from the static counterparts to facilitate evaluating potential approaches. After that, the preliminary analysis for the fitness landscape of the dynamic CARP is conducted by experimental studies.

Yi Mei
School of Computer Science and Information Technology, RMIT University,
Melbourne VIC 3001, Australia
e-mail: `yi.mei@rmit.edu.au`

Ke Tang
Nature Inspired Computation and Applications Laboratory (NICAL),
School of Computer Science, University of Science and Technology of China,
Hefei 230027, China
e-mail: `ketang@ustc.edu.cn`

Xin Yao
Nature Inspired Computation and Applications Laboratory (NICAL), The USTC-Birmingham Joint Research Institute in Intelligent Computation and Its Applications (UBRI), School of Computer Science, University of Science and Technology of China, Hefei 230027, China, and Centre of Excellence for Research in Computational Intelligence and Applications (CERCIA), School of Computer Science, University of Birmingham,
Birmingham B15 2TT, U.K.
e-mail: `x.yao@cs.bham.ac.uk`

15.1 Introduction

The capacitated arc routing problem (CARP) is a classic combinatorial optimization problem that has wide applications in the real world including the salt routing optimization [4, 23, 24, 28, 29, 42], urban waste collection [14, 16, 33, 34, 39] and snow removal [11, 41].

CARP can be described as follows: Given a connected graph, some edges (called the *tasks*) of the graph are required to be served by a vehicle fleet located at the *depot* vertex. The problem aims to determine a least-cost plan subject to the following constraints:

- Each vehicle must start and end at the depot vertex;
- Each task is served exactly once by one vehicle;
- The total demand of the tasks served by each route cannot exceed its capacity.

CARP has been proven to be NP-hard by Golden and Wong in [26]. That is, the computational complexity of finding the global optimum increases exponentially with the increase of problem size. On the other hand, the real-world problems often have quite large problem sizes, making it impractical to find the global optimum by the exact methods. In this situation, methods based on evolutionary computation are promising methods due to their capability of obtaining good sub-optimal solutions within a given time budget.

So far, intensive investigations have been conducted for solving CARP. Most of them are focused on the static CARP, in which all the problem parameters are exactly known in advance and do not change as time goes on. However, in the real world, the above assumption can hardly be guaranteed, and some or all of the problem parameters cannot be known in advance or change over time. For example, in the snow removal application, the amount of snow to be removed for each street cannot be known exactly until the truck finishes the removal, and the traffic jam or road maintenance influences the time needed to traverse a street.

When the problem contains stochastic or dynamic problem parameters, the corresponding CARP can be called the dynamic CARP . Although the dynamic CARP is closer to the reality than the static counterpart, it has been overlooked so far, and there are only a few related research done. Fleury *et al.* proposed a stochastic CARP model in [21]. In the model, the task demands are considered to be stochastic. In this case, a solution that is expected to be feasible may be actually infeasible since the actual total demand served by a route may be larger than expected and exceed the capacity. Therefore, the solution must be made feasible by some repair operator before the calculation of its total cost.

The methods proposed for solving the dynamic CARP are also quite few. Fleury *et al.* proposed an evolutionary algorithm in [21], which employs a simple repair operator and optimizes the total cost of the repaired solution. Christiansen *et al.* proposed a branch-and-price algorithm in [13]. Laporte *et al.* proposed an adaptive large neighborhood search heuristic in [34].

Compared with the dynamic CARP, the stochastic vehicle routing problem (SVRP) , which is the node routing counterpart of the dynamic CARP, has received much more research interest. Starting from the one-vehicle special case of VRP, i.e., the stochastic traveling salesman problem (TSP) , the presence of customers and the travel times between the customers are considered to be stochastic. The TSP with stochastic customers was proposed by Jaillet [31] along with a number of mathematical models, while there has been no mathematical model for the TSP with stochastic travel times. In the *m*-vehicle version of the TSP with stochastic travel times, there are *m* vehicles available instead of only one vehicle. In the problem, all the vehicles have to depart from and arrive at a common depot, and a deadline is imposed on each vehicle route. A penalty is induced for the completion delay. For stochastic VRPs, research works are focused on stochastic demand of customers [5, 18, 45] and on the stochastic presence of customers [5, 47], and both [6]. A comprehensive survey of the aforementioned problems is given in [22], and a dynamic replanning for VRP in case of unforseen situations such as traffic jams is considered in [48] and [49].

In summary, three stochastic factors have been considered in previous research work: (1) the presence of tasks (vertices in VRP and edges in CARP); (2) the demand of the tasks and (3) the deadheading costs (e.g., travel time) between the tasks. In fact, a fourth stochastic factor can be considered: the availability of the path between each pair of vertices. For example, when a street is on its maintenance, it becomes temporarily unavailable to be traversed and thus disappears from the graph. The above four stochastic factors can occur simultaneously in the problem. Unfortunately, a corresponding model has not been investigated. Most research works consider them separately or combine at most two of them together (e.g., the presence and demand of the tasks are combined in [6]). In this chapter, a more general dynamic CARP with all the above four stochastic factors embedded is defined. This dynamic CARP is different from the static CARP with respect to the inputs, outputs, objective and constraints. In the dynamic CARP, the inputs are random variables and the outputs include a solution and a repair operator that can make the solution feasible in any possible environment. The objective of the dynamic CARP switches from obtaining the optimal solution in a specific environment to finding a robust solution, i.e., a solution that shows relatively good performance in all possible environment. The goal of the constraints of the dynamic CARP is no longer to guarantee the feasibility of the solution, but to help improving the robustness of the obtained solution.

After the problem has been defined, preliminary investigations are conducted on it. First, in order to facilitate the potential approaches for the dynamic CARP, the benchmark instances are generated by extending from the static CARP benchmark instances. Then, a rough analysis and discussion about the fitness landscape of the dynamic CARP is conducted. It has been found that for the dynamic CARP, merely using the expected information does not necessarily lead to robust solutions, and the optimal solution for the corresponding static counterpart may be much less robust than other solutions for the dynamic CARP.

The rest of the chapter is organized as follows: First, the dynamic CARP is defined in Section 15.2. After that, the related work for solving the dynamic CARP and its static counterpart is comprehensively reviewed in Section 15.3. Then, the benchmark instances for dynamic CARP is generated in Section 15.4. The preliminary investigation of the fitness landscape of the dynamic CARP is conducted in Section 15.5. Finally, the conclusion and future work is given in Section 15.6.

15.2 Problem Definition

Before defining the dynamic CARP, the simpler and more basic static CARP is first introduced to help understanding. Then, the dynamic CARP is defined and compared with the static CARP.

15.2.1 Static Capacitated Arc Routing Problem

The static CARP is the most basic and simplest version of CARP. It is defined on a weighted connected graph $G(V,E,A)$, where V, E and A are the sets of vertices, undirected edges and directed edges, respectively. For the sake of convenience, the undirect edges will be called *edges* while the directed edges will be called *arcs*. All the edges $(v_i,v_j) \in E$ and arcs $\langle v_i,v_j \rangle \in A$ are associated with a nonnegative serving cost $sc(v_i,v_j)$, a positive deadheading cost $dc(v_i,v_j)$ and a nonnegative demand $d(v_i,v_j)$. The edges and arcs having positive demands are called the *tasks* and must be served. The serving cost of an edge or arc indicates the cost induced by serving it if it is a task, while the deadheading cost indicates the cost induced by traversing it without serving. The sets of edge tasks, arc tasks and total tasks are represented by $E_R = \{(v_i,v_j) \in E | d(v_i,v_j) > 0\}, A_R = \{\langle v_i,v_j \rangle \in A | d(v_i,v_j) > 0\}$ and $R = E_R \cup A_R$. A number of vehicles, each with capacity of Q, are located at the *depot* vertex $v_0 \in V$ to serve the tasks in R.

A CARP solution can be essentially represented by a route set $X = \{X_1,...,X_m\}$ and a corresponding 0-1 vector set $Y = \{Y_1,...,Y_m\}$. The k^{th} route $X_k = (x_{k1},...,x_{kl_k})$, indicating the route traversed by the k^{th} vehicle, is a sequence of vertices starting and ending at v_0, i.e., $x_{k1} = x_{kl_k} = v_0$. The corresponding 0-1 vector $Y_k = (y_{k1},...,y_{k(l_k-1)})$ is defined as follows: if $(x_{ki},x_{k(i+1)})$ is a task and is served at the current position, then $y_{ki} = 1$; otherwise, $y_{ki} = 0$. Under such a solution representation scheme, the static CARP can be stated as follows:

$$\min\ tc(S) = \sum_{k=1}^{m} \sum_{i=1}^{l_k-1} (sc(x_{ki},x_{k(i+1)}) \cdot y_{ki} + dc(x_{ki},x_{k(i+1)}) \cdot (1-y_{ki})) \tag{15.1}$$

$$s.t.:\ x_{k1} = x_{kl_k} = v_0,\ \ k = 1,2,...,m \tag{15.2}$$

$$\sum_{k=1}^{m} \sum_{i=1}^{l_k-1} y_{ki} = |R| \tag{15.3}$$

$$(x_{ki},x_{k(i+1)}) \in E_R \cup A_R,\ \ \forall y_{ki} = 1 \tag{15.4}$$

$$(x_{k_1 i_1},x_{k_1(i_1+1)}) \neq (x_{k_2 i_2},x_{k_2(i_2+1)}),\ \ \forall y_{k_1 i_1} = 1, y_{k_2 i_2} = 1, (k_1,i_1) \neq (k_2,i_2) \tag{15.5}$$

$$(x_{k_1 i_1}, x_{k_1(i_1+1)}) \neq (x_{k_2(i_2+1)}, x_{k_2 i_2}), \ \forall y_{k_1 i_1} = 1, y_{k_2 i_2} = 1, (k_1, i_1) \neq (k_2, i_2) \quad (15.6)$$

$$\sum_{i=1}^{l_k - 1} d(x_{ki}, x_{k(i+1)}) \cdot y_{ki} \leqslant Q, \ k = 1, 2, ..., m \quad (15.7)$$

$$x_{ki} \in V, \ dc(x_{ki}, x_{k(i+1)}) < \infty, \ y_{ki} = 0 \ or \ 1 \quad (15.8)$$

where $|R|$ is the number of tasks in R. In constraints (15.5) and (15.6), the inequality $(k_1, i_1) \neq (k_2, i_2)$ is satisfied if and only if at least one of the two inequalities $k_1 \neq k_2$ and $i_1 \neq i_2$ is satisfied. Objective (15.1) is to minimize the total cost $tc(S)$. Constraint (15.2) indicates all the routes start and end at the depot v_0. Constraints (15.3)–(15.6) guarantee that all the tasks are served exactly once. Constraint (15.7) is the capacity constraint, i.e., the total demands served by each route cannot exceed the capacity Q. Constraint (15.8) defines the domain of the variables.

15.2.2 Dynamic Capacitated Arc Routing Problem

The above static CARP can be characterized by the following four aspects: inputs, outputs, objective and constraints. To be specific, the inputs include the deadheading costs $dc(v_i, v_j)$, the serving costs $sc(v_i, v_j)$, the demand of the tasks $d(v_i, v_j)$ and the capacity Q. The output is a solution represented by a route vector X and a 0-1 vector Y. The objective is to minimize the total cost of the solution. The constraints include the basic constraints (each route starts and ends at the depot and each task is served exactly once) and the capacity constraint. Next, the dynamic CARP is described from the above four aspects as well, and compared with the static CARP with respect to each aspect.

15.2.2.1 Inputs of the Dynamic Capacitated Arc Routing Problem

In the static CARP, all the input parameters are assumed to be known in advantage and fixed over time. As mentioned in Section 15.1, in the real-world applications, this is usually not the case, and it is more appropriate to represent the input parameters as random variables rather than constants.

As mentioned in Section 15.1, there are four stochastic factors in the dynamic CARP: (1) presence of tasks; (2) demand of tasks; (3) presence of paths between vertices and (4) deadheading costs between vertices. These stochastic factors transform the deadheading costs $dc(v_i, v_j)$ and the demand of tasks $d(v_i, v_j)$ from constants to random variables. It should be noted that the serving costs $sc(v_i, v_j)$ are also affected by the stochastic factors. Here, for the sake of simplicity, one can assume that the serving costs are proportional to the deadheading costs, or even equal to them. Then, the two random variables $dc(v_i, v_j)$ and $sc(v_i, v_j)$ can be combined into a single random variable $dc(v_i, v_j)$.

When a task (v_i, v_j) is absent, one can consider that its demand $d(v_i, v_j)$ is zero. Similarly, $dc(v_i, v_j) = \infty$ implies that the edge (v_i, v_j) temporarily disappears.

Therefore, the distribution of the random variables can be seen as a combination of a constant (including infinity) for the case of absence and a random distribution for the case of presence.

In the real-world applications, an implementation process of a solution can be seen as a sample of the random variables. To be specific, during the implementation process, the presence and actual demand of each task is unknown until it has been served, and the presence and deadheading cost between each pair of vertices can only be known after the path has been traversed.

15.2.2.2 Outputs of the Dynamic Capacitated Arc Routing Problem

The stochastic nature of the input parameters in the dynamic CARP influence both the quality and feasibility of solutions. First, it is natural that the change of deadheading costs leads to the change of the total cost, which determines the solution quality. Second, the change of demand or the presence of the tasks that are expected to be absent will make the total demand served by a route larger than expected, and thus violates the capacity constraint. Third, the absence of a path existing in the solution makes the corresponding route disconnected so that the solution becomes illegal unless another path is found to connect the separated vertices. In summary, given a solution, the quality and feasibility changes in the dynamic environment. That is, a solution is feasible in one environment, but is infeasible in another environment. A high-quality solution in one environment may perform quite badly in another environment.

The change of feasibility can only be found during the implementation process. For example, a vehicle can only know whether its remaining capacity can afford the actual demand of the next task after serving it. If the change of feasibility has been detected, i.e., the solution becomes infeasible in the current environment, the solution must be modified (repaired) so that it becomes feasible again. For this purpose, a corresponding repair operator is needed to make the solution feasible whenever it becomes infeasible.

In summary, the outputs of the dynamic CARP should include a solution and a repair operator. The repair operator must be able to make the solution feasible in any possible environment.

15.2.2.3 Objective of the Dynamic Capacitated Arc Routing Problem

It is known that the quality and feasibility of solution are different in different environments. Therefore, it is meaningless to obtain the optimal solution in one specific environment since its performance may severely deteriorate when the environment changes. Instead, the objective of the dynamic CARP should be to obtain a robust solution, i.e., a solution that shows relatively good performance under all possible environments. To this end, a proper robustness measure for the dynamic CARP solutions is to be defined.

Taguchi proposed the concept of robustness optimization for the first time in [43], in which the quality of solution depends on a noise parameter ξ that is out of the control of the designer as well as the control parameter x. To evaluate a solution in this situation, Taguchi defined the following robustness measure:

$$MSD = \frac{1}{k}\sum_{i=1}^{k}(y(x,\xi_i) - \hat{y})^2 \tag{15.9}$$

where $y(x,\xi_i)$ is the actual performance under the control parameter x and the noise ξ_i), k indicates the number of all possible noises, and $\hat{y}$ is the target performance. MSD can be seen as the deviation of the actual performance y of the solution x, which is influenced by the noise ξ, from the target performance $\hat{y}$.

After that, the concept of robustness optimization in uncertain and dynamic environments has received the research interest in various scientific fields such as operations research and engineering design (e.g., [3] [19] [32]), and many other robustness measures have been proposed. Some of the representative measures are introduced below:

1. Worst-case Performance $R_w(x)$**:** Take minimizing the objective function $f(x)$ as an example, the worst-case performance is defined as:

$$R_w(x) = \sup_{x' \in \mathcal{N}(x)} f(x') \tag{15.10}$$

where $\mathcal{N}(x)$ stands for a predefined neighborhood of x. When the actual parameter fluctuates within the neighborhood $\mathcal{N}(x)$ of the solution x due to the stochastic nature, optimizing $R_w(x)$ means optimizing the performance of the solution in the worst case. Examples of the worst-case performance measure can be found in [20] [30] [35] [37].

2. Expected Performance $R_e(x)$**:** The expected performance measures the expectation of $f(x)$ with respect to the environmental parameter ξ. It can be stated as:

$$R_e(x) = E[f(x,\xi)|\xi] = \int f(x;\xi)d\xi \tag{15.11}$$

The expected performance has been adopted in [9] [10] [15] [50] to evaluate the robustness of solutions.

3. Threshold-based Robustness Measure $R_t(x)$**:** In many real-world cases, the objective is not to maximize the performance, but to meet the predefined quality threshold q. In such a situation, one can maximize the probability of reaching the quality threshold, i.e.,

$$R_t(x) = Pr[f(x) \leqslant q] \tag{15.12}$$

4. Reliability-based Robustness Measure: Unlike the above three measures, this measure is used to deal with the stochastic factors that appear in the constraints and change the practical feasibility of solution. Given the following optimization problem:

$$\min\ f(x) \tag{15.13}$$
$$s.t.\ g_i(x) \leqslant 0,\ i = 1,...,I \tag{15.14}$$
$$h_j(x) = 0,\ j = 1,...,J \tag{15.15}$$

in which there is a stochastic constraint $g_k(x;\xi) \leqslant 0$ affected by the environmental parameter ξ. For the same value of x, there may exist ξ_1 and ξ_2 so that $g_k(x;\xi_1) \leqslant 0$ and $g_k(x;\xi_2) > 0$. In other words, x is feasible in environment ξ_1, while becomes infeasible in environment ξ_2. In such a situation, the reliability-based robustness measure transforms the original constraint $g_k(x;\xi) \leqslant 0$ to the following constraint $Pr[g_k(x;\xi) \leqslant 0] \geqslant P_0$, which means the probability of satisfying the constraint is no less than the confidence probability P_0. This measure employs the same idea of the probabilistic constrained programming in stochastic programming [7], and has been widely used in the design optimization problems based on reliability [1, 2, 12, 27, 36, 40].

5. Repair-based Robustness Measure: This measure is used to deal with the stochastic factors appearing in constraints as well. As the name implies, the measure defines a repair operator Φ to make infeasible solutions feasible again. Given a solution x and a sample of the environmental parameter ξ, if x is feasible in the current environment, then it remains unchanged. Otherwise, it is modified by Φ to a feasible solution, i.e., $x \to \Phi(x,\xi)$. Here, the repair operator Φ has to be defined in such a way that for any solution x and environment sample ξ, $\Phi(x,\xi)$ must be a feasible solution. The measure was adopted in [21] in the context of the CARP with stochastic task demands. Under the assumption that the demand of each single task is much smaller than the capacity, the repair operator simply cut the infeasible routes before the last task.

The dynamic CARP has stochastic factors in both the objective function and the constraints. As mentioned above, the stochastic constraints are addressed by the repair operator, and thus the corresponding repair-based robustness measure is to be used. As for the stochastic objective function, measures 1–3 reflect different aspects and can be chosen according to practical consideration.

Then, the robustness measure for the dynamic CARP solutions can be defined as:

$$R(S) = R_*(\Phi(S,\xi)) \tag{15.16}$$

Here, R_* can be R_w, R_e or R_t, and $\Phi(S,\xi)$ is the feasible solution obtained by applying the repair operator Φ on the solution S according to the environment ξ.

15.2.2.4 Constraints of the Dynamic Capacitated Arc Routing Problem

Unlike the static CARP, the feasibility of solution cannot be guaranteed by imposing the constraints since it depends on the actual environment parameters. However, proper constraints can still help find more robust solutions. For example, one can impose a capacity constraint based on the expected demand of tasks so that the obtained solution is expected to satisfy the actual capacity constraint. In practice, different constraints can lead to different optimal solutions in terms of the defined robustness measure.

15.2.2.5 Summary

Table 15.1 summaries the differences between the dynamic CARP and the static CARP with respect to the above four aspects. In the table, "SCARP" and "DCARP" stand for the static and dynamic CARPs, respectively.

Table 15.1 Comparison between the dynamic and static CARPs

Aspect	**SCARP**	**DCARP**
Input	The inputs are constants	The inputs include constants and random variables
Output	A feasible solution in the given specific environment	A solution along with an operator to repair the solution whenever it becomes infeasible
Objective	Minimize the total cost	Optimize the robustness
Constraints	To guarantee the feasibility of solutions	To help obtain more robust solutions

Finally, by setting the robustness measures as R_e and imposing proper constraints, the dynamic CARP can be stated as follows:

$$\min\ R_e(\Phi(S,\xi)) \tag{15.17}$$

$$s.t.:\ x_{k1} = x_{kl_k} = v_0,\ \ k = 1,2,...,m \tag{15.18}$$

$$\sum_{k=1}^{m} \sum_{i=1}^{l_k-1} y_{ki} = |R| \tag{15.19}$$

$$(x_{ki}, x_{k(i+1)}) \in E_R \cup A_R,\ \ \forall y_{ki} = 1 \tag{15.20}$$

$$(x_{k_1 i_1}, x_{k_1(i_1+1)}) \neq (x_{k_2 i_2}, x_{k_2(i_2+1)}),\ \ \forall y_{k_1 i_1} = 1, y_{k_2 i_2} = 1, (k_1, i_1) \neq (k_2, i_2) \tag{15.21}$$

$$(x_{k_1 i_1}, x_{k_1(i_1+1)}) \neq (x_{k_2(i_2+1)}, x_{k_2 i_2}),\ \ \forall y_{k_1 i_1} = 1, y_{k_2 i_2} = 1, (k_1, i_1) \neq (k_2, i_2) \tag{15.22}$$

$$E[\sum_{i=1}^{l_k-1} d(x_{ki}, x_{k(i+1)}, \xi) \cdot y_{ki} | \xi] \leqslant Q,\ \ k = 1,2,...,m \tag{15.23}$$

$$x_{ki} \in V,\ \ dc(x_{ki}, x_{k(i+1)}, \xi) < \infty,\ \ y_{ki} = 0\ \ or\ \ 1 \tag{15.24}$$

Constraints (15.18)–(15.22) and (15.24) are the basic constraints and domain of the variables, while constraint (15.23) implies that the expected total demand of each route does not exceed the capacity.

15.3 Evolutionary Computation for Dynamic Capacitated Arc Routing Problem

The challenges of solving the dynamic CARP come from both the complicatedness of CARP itself and the difficulties caused by the dynamic environment. Next, we will discuss how to address each of the two issues with evolutionary computation, respectively.

15.3.1 Addressing the Capacitated Arc Routing Problem Issues

In this sub-section, two competitive approaches proposed for CARP are introduced, i.e., the Repair-based Tabu Search (RTS) [38] and the Memetic Algorithm with Extended Neighborhood Search (MAENS) [44].

15.3.1.1 The Global Repair Operator and the Repair-Based Tabu Search

The capacity constraint is one of the most important constraints that lead to the complicatedness of CARP. Without the capacity constraint, the problem can be seen as a single routing problem. However, with the capacity constraint imposed, the problem becomes a combination of a routing sub-problem and a clustering sub-problem, both of which are difficult to solve. Therefore, it is important to tackle the capacity constraint properly.

For a solution $S = (X,Y)$, the total cost depends only on the route set X, while the 0-1 vector Y determines whether the solution satisfies the capacity constraint and if not, the extent of the violation. Since a task can be traversed multiple times but only served once in the vertex sequence, a single X can be associated with several different Y's. An example is given in Fig. 15.1. In Fig. 15.1, the vertex o represents the depot. All the 6 edges $\{(o,a),(o,b),(o,c),(a,b),(a,c),(b,c)\}$ are tasks. The capacity of each vehicle is 4. The number on each edge denotes its demand, e.g., $d(a,b) = 3$. The right part of Fig. 15.1 shows two different solutions S_1 and S_2, which share the same route set, but are different in the 0-1 vector. As a result, S_1 is infeasible with one unit capacity violation, i.e., $d(a,b)+d(b,o) = 3+2 = 5 > 4$, whereas S_2 is the global optimum.

During the search process, a low-cost route set and the corresponding feasible 0-1 vector are not easy to be obtained simultaneously, and an infeasible solution is usually discarded because of the bad 0-1 vector despite its promising route set. In order to address this issue, the Global Repair Operator (GRO) [38] is proposed.

Given an infeasible solution, GRO preserves its route set and re-assigns the 0-1 variables to minimize the constraint violation. In other words, GRO seeks the optimal assignment of 0-1 variables for a given route set. Such a repair process

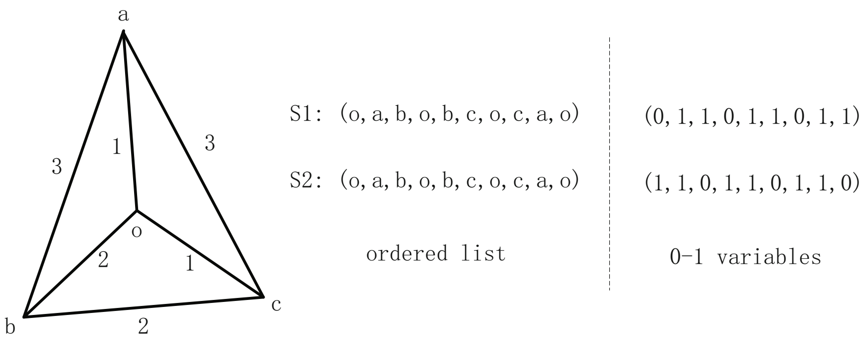

Fig. 15.1 An example of two different solutions sharing the same route set

takes into account all routes involved in the solution, and thus GRO can be viewed as a global operator.

Suppose we have an infeasible solution with m routes, re-assigning the 0-1 variables can be formulated as the following problem:

$$\min \sum_{i=1}^{m} \big(\max\{\sum_{j=1}^{N} s_j x_{ij} - Q, 0\}\big) \tag{15.25}$$

$$s.t.: \sum_{i \in \Omega(j)} x_{ij} = 1, \ \forall j = 1,2,...,N, \tag{15.26}$$

$$x_{ij} = 0 \text{ or } 1, \forall \ i = 1,2,...,m; \ \ j = 1,2,...,N. \tag{15.27}$$

where N is the total number of tasks and s_j denotes the serving cost for task j. x_{ij} is set to 1 if task j is served in the route i, and set to 0 otherwise. $\Omega(j)$ is defined as

$$\Omega(j) = \{\, i \mid \text{task } j \text{ is traversed in route } i \text{ in S}\}$$

Given the vertex sequence, constraints (15.26) and (15.27) guarantee that each task is served only once among the routes it is traversed.

Let $\{a_1, a_2, ..., a_N\}$ and $\{b_1, b_2, ..., b_m\}$ be a set of items and bins, respectively. The above problem can be viewed as a bin-packing problem, where the size of the item j is s_j, and all the bins share an identical capacity Q. GRO employs an insertion heuristic followed by a short-term tabu search to solve the bin-packing problem. The general idea of the insertion heuristic is straightforward. We sequentially pick an item out of the whole set and insert it into a bin, until all the items have been inserted. Each item is inserted in a bin that minimizes the objective function (15.25). Such procedure can be described as follows:

Step 1: Initialize $x_{ij} = 0, \forall i, j$. Let $A = \{1, 2, ..., N\}$ and $cl(b_i) = 0, \forall i$. Here, $cl(b_i)$ is the current load of b_i. Then repeat step 2 to step 4 until $A = \emptyset$.

Step 2: For each $j \in A$, identify the set $\Omega'(j)$ satisfying $\Omega'(j) = \{i \in \Omega(j) | cl(b_i) + s_j \leqslant Q\}$. Select the item corresponding to the smallest $|\Omega'(j)|$ as the one to be inserted. If multiple items share the smallest $|\Omega'(j)|$, the one with the largest s_j will be selected. Then ties are broken by selecting the item with the smallest index (j). The selected index is j^*. Selecting the items in this way guarantees that the item with the least choice of insertion without violating the constraints is chosen first.

Step 3: Identify the b_i with the smallest $cl(b_i)$ from $\Omega'(j^*)$. If more than one bin has the smallest $cl(b_i)$, the one with the smallest

$$\sum_{j \in A} I_{\Omega(j)}(i) \cdot s_j$$

is selected, where $I_{\Omega(j)}(i)$ is an indicator function. $I_{\Omega(j)}(i) = 1$ if $i \in \Omega(j)$, and 0 otherwise. The above equation indicates that the bin available for the least untreated items is considered first. After that, ties are broken by selecting the bin with the smallest index (i). The selected index is i^*.

Step 4: Insert the selected item a_{j^*} in the chosen bin b_{i^*}. Set $x_{i^* j^*} = 1$, remove j^* from A and update $cl(b_{i^*})$ with s_{j^*}.

After obtaining the initial solution with the insertion heuristic, a standard tabu search is employed to further improve it. The tabu search is described in Algorithm 1. S_0 is the solution obtained by the insertion heuristic, and $f(S)$ is the objective function (15.25). The neighborhood $N(S)$ of solution S indicates the set of solutions that can be obtained by moving an item to another admissible bin. The tabu list is designed as follows: when an item is moved from one bin to another, it is not allowed to be moved back to its original bin in a certain number of subsequent iterations, unless the movement leads to a better solution than the current best solution. Here, the tabu tenure is set to $F/2$, where F is the number of items with more than one admissible bin ($|\Omega(j)| > 1$). The tabu search process terminates after N iterations or $N/2$ consecutive iterations without improvement.

Based on a solution of the bin-packing problem, a new solution of CARP can be directly obtained by updating the 0-1 variables according to x_{ij}'s. However, the tabu search process might generate multiple assignments of 0-1 variables that all correspond to feasible solutions of the CARP, which are saved in the archive A (lines 9–11 of Algorithm 1). A further refinement procedure is needed to select the best solution in A.

In the insertion heuristic and tabu search process, the total cost is not considered since the route set is assumed to be unchanged. However, after the change of 0-1 variables, the adjacent services may be connected with shorter paths. Hence, as the final step of the GRO, all the archived assignments of 0-1 variables are transformed to the corresponding solutions of CARP, and then the route sets of these solutions are refined by updating the vertices between each pair of adjacent services with the shortest path. Finally, the solution with the smallest cost is chosen as the output of the GRO.

To summarize, the major steps of the GRO are listed as follows:

1. Formulate the repair operation as a bin-packing problem and get a solution via the insertion heuristic;

Algorithm 1 A = TS(f, S_0)

```
1: Set the current solution $S = S_0$, the current best solution $S_b = S_0$. Set $A = \emptyset$;
2: while the stopping criteria are not satisfied do
3:     Set $f(S') = \infty$;
4:     for all $S'' \in N(S)$ do
5:         if $S''$ is not tabu and $f(S'') < f(S')$ then
6:             $S' = S''$, $f(S') = f(S'')$;
7:         end if
8:     end for
9:     if $f(S') = 0$ then
10:        $A = A \cup S'$;
11:    end if
12:    if $f(S') < f(S_b)$ then
13:        $S_b = S'$, $f(S_b) = f(S')$;
14:    end if
15:    Update the tabu list and set $S = S'$;
16:    if $A = \emptyset$ then
17:        $A = \{S_b\}$;
18:    end if
19: end while
20: return $A$;
```

2. Utilize a tabu search process to further improve the solution obtained in the first step, and get an archive of candidate assignments of 0-1 variables.
3. Obtain new solutions of CARP based on the archived assignments of 0-1 variables and update these solutions with the further refinement procedure. The solution with the smallest cost is chosen as the output.

The GRO can be easily embedded in any search-based approach to enhance its search capability. The RTS is thus proposed by simply embedding the GRO into an existing competitive tabu search algorithm [8]. Specifically, for each infeasible solution with promising total cost (which is smaller than the total cost of the best feasible solution found so far), the GRO is applied to reduce its violation to the capacity constraint.

15.3.1.2 Memetic Algorithm with Extended Neighborhood Search

Another difficulty of CARP is that the existing search operators are with small step sizes, and thus have difficulty to explore in the large solution space. In this situation, a search operator with large step size is more desirable. However, it is not a trivial task to design such a search operator. An intuitive idea is to apply the traditional search operators for multiple times. Nevertheless, the neighborhood size increases exponentially with the number of times to apply the search operators. As a result, it is prohibitive to enumerate all the possible solutions in the neighborhood. One simple solution to this problem is to randomly sample a part of the huge neighborhood. However, it is often the case that some regions in the solution space are more promising than the others. Hence, random sampling is a bit blind and might waste a lot of computational resource. To summarize, although a large step-size local search can be beneficial, it cannot be implemented by simply extending the traditional move operators, and a more refined approach is required. For this purpose, a Merge-Split (MS) operator [44] is developed.

The MS operator aims to improve a given solution by modifying multiple routes of it. As indicated by its name (Merge-Split), this operator has two components, i.e., the Merge and Split components. Given a solution, the Merge component randomly selects p ($p > 1$) routes of it, combines them together to form an unordered list of tasks, which contains all the tasks of the selected routes. The Split component directly operates on the unordered list generated by the Merge component, which is composed of the path scanning (PS) heuristic [25] and Ulusoy's split procedure [46]. Given a set of tasks and the whole graph, PS is used to quickly generate a set of feasible routes that serve all the given tasks with a relatively low total cost. It starts by initializing an empty path. At each iteration, it seeks the tasks that do not violate the capacity constraint. If no task satisfies the constraint, it connects the end of the current path to the depot with the shortest path between them to form a route, and then initializes a new empty path. If a unique task satisfies the constraint, PS connects that task to the end of the current path (again, with the shortest path between them). If multiple tasks satisfy the constraint, the one closest to the end of the current path is chosen. If multiple tasks not only satisfy the capacity constraint, but also are the closest to the end of the current path, five rules are further adopted to determine which to choose: (1) maximize the distance from the head of task to the depot; (2) minimize the distance from the head of task to the depot; (3) maximize the term $d(t)/sc(t)$, where $d(t)$ and $sc(t)$ are demand and serving cost of task t, respectively; (4) minimize the term $d(t)/sc(t)$; (5) use rule (1) if the vehicle is less than half-full, otherwise use rule (2). If multiple tasks still remain, ties are broken arbitrarily. PS terminates when all the tasks in the unordered list have been selected. Note that PS does not use the five rules alternatively. Instead, it scans the unordered list of tasks for five times. In each scan, only one rule is used. Hence, PS will generate five route sets in total. Then, Ulusoy's split procedure is applied to all the five route sets to further improve them. Here, the route sets can be seen as an ordered list of tasks, and Ulusoy's split procedure can obtain the optimal feasible route sets for the ordered list of tasks.

To summarize, the MS operator first merges multiple routes to obtain an unordered list of tasks, then employs PS to sort the unordered list. After that, Ulusoy's splitting procedure is used to split the ordered lists into new routes in the optimal way. Finally, we may obtain five new solutions of CARP by embedding the new routes back into the original solution, and the best one is chosen as the output of the MS operator. Figure 15.2 demonstrates the whole process of the MS operator.

The advantages of the MS operator are twofold. First, it can generate new solutions that are significantly different from the current one as it conducts on routes instead of tasks. In general, the larger the p (i.e., the number of routes involved in MS), the more distant the new solution is from the current solution. Second, the new solutions obtained by the MS operator tend to have low total cost due to the

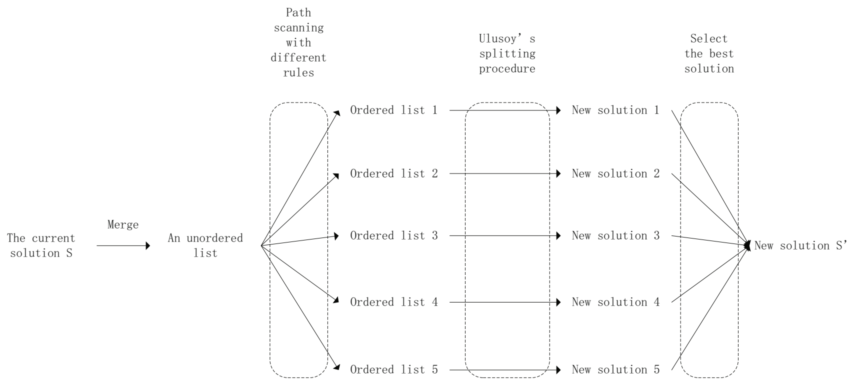

Fig. 15.2 The process of the MS operator

Algorithm 2 The brief description of MAENS

```
1: **Initialization:** Generate an initial population;
2: **while** Stopping criteria are not satisfied **do**
3:     Select the parent solutions and generate offsprings by the crossover operator;
4:     **for** each offspring **do**
5:         Perform extended neighborhood search around it with probability $P_{ls}$;
6:         Select solutions from the original ones and the offsprings to form the population in
           the next generation.
7:     **end for**
8: **end while**
```

adoption of PS and Ulusoy's splitting procedure, both of which are known to be capable of generating relatively good solutions. On the other hand, the major drawback is its high computational complexity. Fortunately, such a drawback may be more or less alleviated by a careful coordination of the MS operator and other search operators. For this purpose, the memetic algorithm framework is adopted and the MS and traditional move operators are integrated to form the local search with extended neighborhood. The resultant algorithm is thus called the Memetic Algorithm with Extended Neighborhood Search (MAENS). A brief description of MAENS is given in Algorithm 2.

During the extended neighborhood search process, the MS and traditional search operators are employed in the following way: Given an offspring individual generated by the crossover operator, the traditional move operators (i.e., the single insertion, double insertion and swap) are applied to the individual until the local optimum is reached. After that, the MS operator is applied to this local optimal solution to form the second stage of the local search, and the local optimum with respect to the extended neighborhood is obtained. Finally, the traditional-neighborhood-based local search is again applied to further refine the local optimum obtained in the second stage and exploit the new local region.

It has been demonstrated in [38] and [44] that the RTS and MAENS are both competitive approaches for CARP. RTS can obtain solutions as well as the best-known ones in a shorter time than other state-of-the-art methods, while MAENS is able to find better solutions than the best-known ones at the cost of more computational efforts.

15.3.2 Tackling the Dynamic Environment

The dynamic environment changes the actual feasibility of solutions. As mentioned before, a repair operator must be designed to make the solution feasible whenever it becomes infeasible in the current environment. An ideal repair operator should not only be able to repair any infeasible solution, but also be practical. In many real-world applications, the departments in charge wish to keep the modification as small as possible during the repair process. Based on such practical consideration, the repair operator Φ should satisfy the following conditions:

- If all the constraints are satisfied, then Φ should not make any change on the solution;
- If the solution violates the capacity constraint, then Φ should cut the infeasible routes into several feasible routes;
- If the path between adjacent vertices in the solution becomes absent, then Φ connects them with another path in the graph.

All the above operations minimizes the modification of the solution.

The sampling values of the random variables can only be known during the implementation process. To be specific, the vehicle does not know whether the path from vertex x_{ki} to its successor $x_{k(i+1)}$ is connected until it reaches x_{ki}. The deadheading cost of $(x_{ki}, x_{k(i+1)})$ can only be known after the vehicle has arrived at $x_{k(i+1)}$. Besides, if $(x_{ki}, x_{k(i+1)})$ is a task, its demand cannot be known until the vehicle finishes its service and reaches $x_{k(i+1)}$. In this situation, the repair operation of Φ can be defined as follows: Given a solution $S = (\{X_1, ..., X_m\}, \{Y_1, ..., Y_m\})$, each vehicle starts from $x_{k1} = v_0$ and traverses according to the vertex sequence $X_k = (x_{k1}, ..., x_{kl_k})$. Once the vehicle arrives at a new vertex (including the starting vertex v_0), Φ checks the value of the first time occurring $y_{kj} = 1$ in the sub-vector $Y_k = (y_{ki}, ..., y_{k(l_k-1)})$ to locate the position of the next task $(x_{kj}, x_{k(j+1)})$. If there is no task left, then it returns the depot through the predefined path. Otherwise, Φ examines whether the current residue is enough to serve the expected demand $E[d(x_{kj}, x_{k(j+1)}, \xi)|\xi]$. If so, then the vehicle traverses to x_{kj}. Otherwise, the vehicle returns to the depot and update the capacity, then goes to x_{kj} again. Note that the practical demand of $(x_{kj}, x_{k(j+1)})$ may be larger than expected and exhaust the capacity on the way of its service. In this case, the vehicle neglects the remaining demand and returns to the depot to update the capacity, and then goes back to x_{kj} and finish the service of $(x_{kj}, x_{k(j+1)})$. All the above repair operations are done through the shortest paths between the origin and the target. In this way, the satisfactory of the capacity constraint can be guaranteed. On the other hand, if it is found that the path from the current vertex x_{ki} to the next vertex $x_{k(i+1)}$ disappears, Φ simply update the deadheading cost matrix by setting $dc(x_{ki}, x_{k(i+1)}) = \infty$, and calculate the shortest path under the updated cost matrix by Dijstra's alorithm [17]. Then, it replaces the interrupted path with the new shortest path.

Based on the above descriptions, Φ can be divided into a repair operator Φ_d to deal with stochastic task demands and a repair operator Φ_c to deal with stochastic presence of paths. Given an infeasible solution, it is first repaired by Φ_d, and then repaired by Φ_c. Algorithms 3 and 4 give the pseudo codes of the repair operators Φ_d and Φ_c, respectively.

In Algorithms 3 and 4, the function $X.push(a)$ indicates inserting the element a (can be a single element or a sequence) into the end of the sequence X, and $|X|$ stands for the length of X. 0_l represents the sequence composed with l 0's. $ESP(v_i, v_j)$ is the shortest path from v_i to v_j under the expected deadheading cost matrix, while $SP(v_i, v_j, \xi)$ is that under the practical deadheading cost matrix of the environmental parameter ξ.

Under the definition of Φ_d and Φ_c, an infeasible solution S is repaired at the following two steps: 1) $S' = \Phi_d(S)$; 2) $S^{\xi} = \Phi_c(S')$.

15.4 Benchmark for Dynamic Capacitated Arc Routing Problem

In order to evaluate the performance of potential approaches, benchmark instances of the dynamic CARP are needed. However, there has been no benchmark instance proposed so far. In [21], the well-known *gdb* test set with static CARP instances was extended to a stochastic CARP test set by replacing the deterministic demand $d(t_i)$ of each task t_i with a Gaussian distributed random variable $D(t_i) \sim N(d(t_i), \sigma_i^2)$, where the variance $\sigma_i = k \times d(t_i)$ is proportional to $d(t_i)$. Here, we generate the benchmark instances for the dynamic CARP similarly. In contrast with [21], we choose Gamma distribution instead of normal distribution. The reason is that all the random variables in the dynamic CARP is nonnegative, and Gamma distribution is one of the most commonly used distribution with nonnegative support set for simulating real environments. Besides, $d(v_i, v_j)$ equals zero with probability $1 - p_{ij}$, and $dc(v_i, v_j)$ equals infinity with probability $1 - p_{ij}$. Hence, in the dynamic CARP, the random variables should satisfy the following distributions:

$$D(v_i, v_j)\begin{cases} \sim G(k_{ij}^d, \theta_{ij}^d), & r < p_{ij}; \\ = 0, & \text{otherwise.} \end{cases} \tag{15.28}$$

$$DC(v_i, v_j)\begin{cases} \sim G(k_{ij}^c, \theta_{ij}^c), & r < q_{ij}; \\ = \infty, & \text{otherwise.} \end{cases} \tag{15.29}$$

where $G(k, \theta)$ is the Gamma distribution with the shape parameter k and the scale parameter θ. The probability density function of $G(k, \theta)$ is $pdf(x; k, \theta) = x^{k-1} \frac{e^{-x/\theta}}{\theta^k \Gamma(k)}$ for $x > 0$ and $k, \theta > 0$, where $\Gamma(k) = \int_0^{\infty} t^{k-1} e^t dt$. It is known that the mean of the Gamma distribution $G(k, \theta)$ is $\mu = k\theta$ and $G(k, \theta)$ converges to the

Algorithm 3 $S' = \Phi_d(S, \xi)$

```
1:  for $k = 1 \to m$ do
2:      Set $X'_k = (v_0)$, $Y'_k = ()$, $\Delta Q = Q$;
3:      for $i = 1 \to |X_k| - 1$ do
4:          if $y_{ki} = 0$ then
5:              $X'_k.push(x_{k(i+1)})$, $Y'_k.push(0)$;
6:          else
7:              if $d(x_{ki}, x_{k(i+1)}, \xi) = 0$ then
8:                  $X'_k.push(x_{k(i+1)})$;
9:                  $Y'_k.push(0)$;
10:             else if $\Delta Q < d(x_{ki}, x_{k(i+1)}, \xi)$ then
11:                 $X'_k.push(ESP(x_{k(i+1)}, v_0))$;
12:                 $X'_k.push(v_0)$;
13:                 $Y'_k.push(1)$;
14:                 $Y'_k.push(0_{|ESP(x_{k(i+1)}, v_0)|})$;
15:                 $X'_k.push(ESP(v_0, x_{ki}))$;
16:                 $Y'_k.push(0_{|ESP(v_0, x_{ki})|})$;
17:                 $X'_k.push(x_{ki})$, $Y_k.push(0)$;
18:                 $\Delta Q \leftarrow \Delta Q + Q - d(x_{ki}, x_{k(i+1)}, \xi)$;
19:             else
20:                 $X'_k.push(x_{k(i+1)})$, $Y'_k.push(1)$, $\Delta Q \leftarrow \Delta Q - d(x_{ki}, x_{k(i+1)}, \xi)$;
21:             end if
22:             for $j = i + 1 \to |X_k| - 1$ do
23:                 if $Y_{kj} = 1$ then
24:                     break;
25:                 end if
26:             end for
27:             if $Y_{kj} = 1 \& \Delta Q < E[d(x_{kj}, x_{k(j+1)}, \xi)|\xi]$ then
28:                 $X'_k.push(ESP(x_{k(i+1)}, v_0))$;
29:                 $X'_k.push(v_0)$;
30:                 $Y'_k.push(0_{|ESP(x_{k(i+1)}, v_0)|+1}$;
31:                 $X'_k.push(ESP(v_0, x_{kj}))$;
32:                 $X'_k.push(x_{kj})$;
33:                 $Y'_k.push(0_{|ESP(v_0, x_{kj})|+1})$;
34:                 $i \leftarrow j$, $\Delta Q \leftarrow Q$;
35:             end if
36:         end if
37:     end for
38: end for
39: return $S' = (\{X'_1, ..., X'_m\}, \{Y'_1, ..., Y'_m\})$;
```

Gaussian distribution when the shape parameter k becomes infinite. According to the idea of the dynamic CARP instance generation in [21], the random variables in the dynamic CARP should have the following properties:

Property 1: The Gamma distribution is close to Gaussian distribution;
Property 2: The expected value of the random variables equals their static values;

Algorithm 4 $S^{\xi} = \Phi_c(S', \xi)$

```
1: for $k = 1 \to m$ do
2:     Set $X_k^{\xi} = (v_0), Y_k^{\xi} = ()$;
3:     for $i = 1 \to |X'_k| - 1$ do
4:         if $dc(x'_{ki}, x'_{k(i+1)}, \xi) < \infty$ then
5:             $X_k^{\xi}.push(x'_{k(i+1)}), Y_k^{\xi}.push(y'_{ki})$;
6:         else
7:             $X_k^{\xi}.push(SP(x'_{ki}, x'_{k(i+1)}, \xi)), X_k^{\xi}.push(x'_{k(i+1)})$;
8:             $Y_k^{\xi}.push(0_{|SP(x'_{ki}, x'_{k(i+1)}, \xi)|+1})$;
9:         end if
10:    end for
11: end for
12: return $S^{\xi} = (\{X_1^{\xi}, \ldots, X_m^{\xi}\}, \{Y_1^{\xi}, \ldots, Y_m^{\xi}\})$;
```

Property 1 can be realized by setting a sufficiently large k. For Property 2, on the other hand, the realizations of the stochastic demands and deadheading costs are different. For the task demand $D(v_i, v_j)$, Property 2 can be realized by directly setting

$$E[D(v_i, v_j)] = p_{ij} k_{ij}^d \theta_{ij}^d + (1 - p_{ij}) \times 0 = d(v_i, v_j) \tag{15.30}$$

However, the above equation is not available for the deadheading cost $DC(v_i, v_j)$ since it is likely to become infinity. Therefore, we neglect such case and only set

$$E[DC(v_i, v_j)] = k_{ij}^c \theta_{ij}^c = dc(v_i, v_j) \tag{15.31}$$

In practice, the shape parameters k_{ij}^d and k_{ij}^c are set to 20. Figure 15.3 gives the probability density functions of the Gamma distribution with $k = 20$ and $\theta = 1.0, 1.5, 2.0$. The settings of θ_{ij}^d and θ_{ij}^c can be derived from Eq. (15.30) and (15.31) as follows:

$$\theta_{ij}^d = \frac{d(v_i, v_j)}{p_{ij} k_{ij}^d} \tag{15.32}$$

$$\theta_{ij}^c = \frac{dc(v_i, v_j)}{k_{ij}^c} \tag{15.33}$$

Finally, p_i and q_{ij} can be intuitively set to 0.9 and 0.95, respectively.

The C++ source code of the instance generator for extending the static CARP instances to the dynamic CARP instances can be downloaded from the website at http://goanna.cs.rmit.edu.au/∼e04499, along with the dynamic CARP instances, namely the *Dgdb*, *Dval* and *Degl* sets, respectively. For each static instance, 30 sampling instances were generated one by one by the instance generator with the starting random seed of 0. If necessary, users can also generate more samplings with other random seeds.

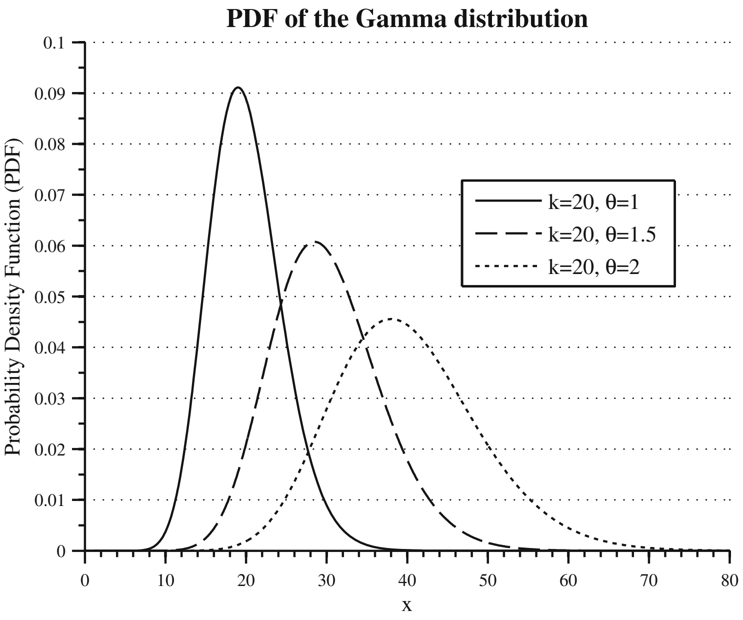

Fig. 15.3 Probability density functions of Gamma distribution with $k = 20$ and $\theta = 1.0, 1.5, 2.0$

15.5 Preliminary Investigation of the Fitness Landscape

The stochastic characteristic of the random variables makes the fitness landscape of the dynamic CARP much more complicated than the static CARP. To investigate the fitness landscape of the dynamic CARP and the impact of the stochastic factors on the performance of the search algorithms, the two static CARP approaches introduced in Section 15.3.1, i.e., RTS and MAENS, were applied to the *Dgdb* set, which is the simplest and smallest test set among the three dynamic CARP benchmark sets generated in Section 15.4. In this way, it is easier to observe how the performance of the algorithms are influenced by the dynamic environment, but not the complicatedness of the problem itself. As demonstrated in [38] and [44], the two selected algorithms are able to reach the global optima for all the static version of the *Dgdb* instances, i.e., the *gdb* instances.

In the experiments, RTS and MAENS were implemented once on all the *gdb* instances, and the sequences of best feasible solutions updated during the search process were recorded. Recalling that the expected values of the random variables of the *Dgdb* instances are equal to the corresponding static values of the *gdb* instances. Thus, by solving the *gdb* instances, the algorithms can be seen as solving the *Dgdb* instances by utilizing the expectation of the random variables. For each solution recorded in the best feasible solution sequence obtained by RTS and MAENS, denoted as $(S_{11}, ..., S_{1l_1})$ and $(S_{21}, ..., S_{2l_2})$, the robustness $R(S_{ij})$ is calculated in terms of the average total cost of the 30 corresponding *Dgdb* instance samples generated in Section 15.4, i.e.,

$$R(S_{ij}) = \frac{1}{30}\sum_{k=1}^{30} tc(S_{ij,\xi_k}) \tag{15.34}$$

where $S_{ij,\xi_k} = \Phi(S_{ij}, \xi_k)$ is the feasible solution obtained by applying Φ to S_{ij} under ξ_k, which is the environmental parameter of the k^{th} sample.

Table 15.2 presents the experimental results. In the table, the columns headed "tcBest" and "RBest" stand for the solutions with the lowest $tc(S)$ and the lowest $R(S)$ among all the recorded solutions. The column headed "RBK" presents the best-known solution with respect to $R(S)$. The columns headed "$tc(S)$" and "$R(S)$" are the objective functions defined for the static and uncertain versions, respectively. $tc(S)$ is defined in Eq. (15.1) and $R(S)$ is defined in Eq. (15.16). For $tc(S)$, the optimal values are marked in bold. As mentioned before, MAENS and RTS can both reach the optimal solutions for the static version of the instances. Therefore, the best $tc(S)$ obtained by them were marked in bold for all the instances.

From Table 15.2, it is observed that $R(S)$ is not proportional to $tc(S)$. For MAENS, the solution with the lowest $tc(S)$ and the solution with the lowest $R(S)$ are different on 9 out of the total 23 instances. For RTS, such a phenomenon occurs also on 9 instances. Another interesting observation is that for the lowest $tc(S)$ obtained by MAENS and RTS, although their values are the same and optimal, the corresponding $R(S)$ of the solution can be very different (e.g., in *Dgdb*12, the solutions with lowest $tc(S)$ obtained by the two algorithms have their $R(S)$'s of 642.03 and 603.54, respectively). Based on the above observation, one can conclude that for a static CARP instance, there often exist multiple global optima. However, their robustness in the corresponding dynamic versions may be quite different. When looking at the best-known solutions with respect to $R(S)$, it is seen that for 16 out of the total 23 instances, the best-known solutions have non-optimal $tc(S)$'s. This implies that the global optimum (in terms of robustness) in a dynamic CARP instance may be quite far away from the global optimum in its static counterpart. Comparing with the results obtained by MAENS and RTS, the $R(S)$ values of the best-known solutions are much smaller than the lowest $R(S)$'s obtained by the two algorithms, not to mention the $R(S)$ of the solutions with lowest $tc(S)$. Therefore, we can conclude that when solving *Dgdb* instances by applying algorithms to the *gdb* counterparts, it is difficult to achieve highly robust solutions.

One possible reason that the algorithms for the static CARP cannot perform well when applied to the dynamic CARP may be explained as follows: the algorithms ignore the possibility that the routes may be cut at certain intermediate positions due to the violation of the capacity constraint during the implementation. If the cut position is distant from the depot, the repaired solution will have a much larger total cost. Since the cut position depends on the allocation of the task services, one solution to address this issue is to modify the allocation of the task services so that the possible cut positions are close to the depot.

In summary, the following observations can be drawn:

Table 15.2 The experimental results of MAENS and RTS on the *Dgdb* set. The optimal $tc(S)$'s are marked in bold.

Name	MAENS				RTS				RBK	
	tcBest		RBest		tcBest		RBest			
	$tc(S)$	$R(S)$	$tc(S)$	$R(S)$	$tc(S)$	$R(S)$	$tc(S)$	$R(S)$	$tc(S)$	$R(S)$
1	**316**	380.63	**316**	380.63	**316**	401.95	323	387.32	323	349.49
2	**339**	436.69	345	401.24	**339**	417.96	**339**	417.96	353	383.72
3	**275**	331.19	**275**	331.19	**275**	323.81	**275**	323.81	296	307.00
4	**287**	350.40	**287**	350.40	**287**	345.77	**287**	345.77	**287**	328.32
5	**377**	492.38	383	472.44	**377**	492.00	**377**	492.00	395	437.79
6	**298**	353.69	**298**	353.69	**298**	369.63	310	367.10	319	342.18
7	**325**	380.06	**325**	380.06	**325**	400.50	**325**	400.50	**325**	356.09
8	**348**	470.18	354	456.70	**348**	464.33	356	449.56	362	443.87
9	**303**	404.34	309	391.77	**303**	394.75	**303**	394.75	337	385.86
10	**275**	306.72	**275**	306.72	**275**	325.48	**275**	325.48	283	291.61
11	**395**	431.86	**395**	431.86	**395**	442.19	**395**	442.19	409	419.06
12	**458**	642.03	468	595.33	**458**	603.54	468	601.58	474	587.33
13	**536**	598.03	554	593.66	**536**	623.52	552	603.43	544	569.82
14	**100**	118.44	**100**	118.44	**100**	118.47	**100**	118.47	**100**	107.90
15	**58**	60.76	**58**	60.76	**58**	58.91	**58**	58.91	**58**	58.09
16	**127**	146.53	129	145.73	**127**	146.97	**127**	146.97	129	133.43
17	**91**	94.36	**91**	94.36	**91**	96.17	**91**	96.17	**91**	92.32
18	**164**	180.75	**164**	180.75	**164**	181.48	168	179.16	**164**	170.90
19	**55**	67.49	**55**	67.49	**55**	64.24	**55**	64.24	**55**	63.04
20	**121**	139.06	125	135.89	**121**	141.43	122	134.82	123	126.01
21	**156**	171.60	**156**	171.60	**156**	177.74	158	174.79	158	165.41
22	**200**	217.01	**200**	217.01	**200**	218.59	202	217.35	204	210.17
23	**233**	263.60	235	256.52	**233**	260.97	**233**	260.97	235	252.35

- The robustness of solution in the dynamic CARP is conflicting with the absolute performance, and the two measures can hardly reach optimum at the same time;
- Static CARP instances often have multiple global optimal solutions. However, their robustness in the corresponding dynamic version may be quite different;
- Only utilizing the expected information cannot lead to highly robust solutions.
- The algorithms for the static CARP cannot perform well for the dynamic CARP because the possible cut position and the additional cost induced by the cut is not considered. To address this issue, one can estimate the probability of the cut position and adjust the task services so as to reduce the expected additional cost caused by the cut.

15.6 Conclusion

In this chapter, a general dynamic CARP is defined and investigated with evolutionary computation. The dynamic CARP model includes the following four stochastic factors: (1) presence of tasks; (2) demand of tasks; (3) presence of paths between vertices and (4) deadheading costs between vertices. These stochastic factors exist in both the objective and constraints of the problem, and thus influence both the performance and feasibility of solution. Unlike the static CARP the outputs of the dynamic CARP include a solution and a repair operator to modify the solution so that all the constraints can be satisfied during the implementation process. Besides, the objective of the dynamic CARP is to optimize the robustness rather than the absolute quality in a specific environment.

The dynamic CARP has the difficulties caused by the complicatedness of CARP and the dynamic environment. To address the two issues, two competitive approaches for CARP, i.e., RTS and MAENS, are introduced, and a repair operator is designed under the practical considerations.

Then, to investigate the fitness landscape of the dynamic CARP, RTS and MAENS were tested on the *Dgdb* benchmark set, which is extended from the *gdb* static CARP benchmark set. It is found that, although the two algorithms showed excellent performance for static CARP, they were not able to find robust solutions for the dynamic CARP. Therefore, the future work is to design new algorithms that can find more robust solutions by taking advantage of more information. One possible direction is to select the solutions in which the adjacent tasks can be connected by multiple paths with nearly the same lengths to avoid the additional cost induced by the absence of edges.

Although no effective approach has been proposed for the dynamic CARP in this chapter, the formal definition of the problem and the generated benchmark provide a solid foundation of further research work, and the analysis and discussions about the problem characteristics give some guidelines for the algorithm design. The future work includes developing algorithms by taking these analytical results into account.

Acknowledgements. This work was supported by the National Natural Science Foundation of China (NSFC) grant No. 61028009.

References

[1] Agarwal, H.: Reliability based design optimization: Formulations and methodologies. Ph.D. thesis, University of Notre Dame, South Bend, IN, USA (2004)

[2] Agarwal, H., Renaud, J.: Reliability based design optimization using response surfaces in application to multidisciplinary systems. Engineering Optimization 36(3), 291–311 (2004)

[3] Allen, J., Seepersad, C., Choi, H., Mistree, F.: Robust design for multiscale and multidisciplinary applications. Journal of Mechanical Design 128(4), 832–843 (2006)

[4] Amberg, A., Domschke, W., Voß, S.: Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees. European Journal of Operational Research 124(2), 360–376 (2000)

[5] Bertsimas, D.: Probabilistic combinatorial optimization problems. Ph.D. thesis, Massachusetts Institute of Technology, Department of Mathematics (1988)

[6] Bertsimas, D.: A vehicle routing problem with stochastic demand. Operations Research, 574–585 (1992)

[7] Birge, J., Louveaux, F.: Introduction to stochastic programming. Springer (1997)

[8] Brandão, J., Eglese, R.: A deterministic tabu search algorithm for the capacitated arc routing problem. Computers and Operations Research 35(4), 1112–1126 (2008)

[9] Branke, J.: Creating robust solutions by means of evolutionary algorithms. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 119–128. Springer, Heidelberg (1998)
[10] Branke, J.: Evolutionary optimization in dynamic environments. Kluwer Academic Pub. (2002)
[11] Campbell, J., Langevin, A.: Roadway snow and ice control. In: Arc Routing: Theory, Solutions and Applications, pp. 389–418. Kluwer, Boston (2000)
[12] Chan, K., Skerlos, S., Papalambros, P.: Monotonicity and active set strategies in probabilistic design optimization. Journal of Mechanical Design 128(4), 893–900 (2006)
[13] Christiansen, C., Lysgaard, J., Wøhlk, S.: A Branch-and-Price Algorithm for the Capacitated Arc Routing Problem with Stochastic Demands. Operations Research Letters 37(6), 392–398 (2009)
[14] Chu, F., Labadi, N., Prins, C.: A scatter search for the periodic capacitated arc routing problem. European Journal of Operational Research 169(2), 586–605 (2006)
[15] Das, I.: Robustness optimization for constrained nonlinear programming problems. Engineering Optimization 32(5), 585–618 (2000)
[16] De Rosa, B., Improta, G., Ghiani, G., Musmanno, R.: The arc routing and scheduling problem with transshipment. Transportation Science 36(3), 301–313 (2002)
[17] Dijkstra, E.: A note on two problems in connexion with graphs. Numerische Mathematik 1(1), 269–271 (1959)
[18] Dror, M., Laporte, G., Trudeau, P.: Vehicle routing with stochastic demands: Properties and solution frameworks. Transportation Science 23(3), 166–176 (1989)
[19] Du, X., Wang, Y., Chen, W.: Methods for robust multidisciplinary design. Tech. rep., American Institute of Aeronautics and Astronautics (2000)
[20] El Ghaoui, L., Lebret, H.: Robust solutions to least-squares problems with uncertain data. SIAM Journal on Matrix Analysis and Applications 18(4), 1035–1064 (1997)
[21] Fleury, G., Lacomme, P., Prins, C.: Evolutionary algorithms for stochastic arc routing problems. In: Raidl, G.R., et al. (eds.) EvoWorkshops 2004. LNCS, vol. 3005, pp. 501–512. Springer, Heidelberg (2004)
[22] Gendreau, M., Laporte, G., Séguin, R.: Stochastic vehicle routing. European Journal of Operational Research 88(1), 3–12 (1996)
[23] Ghiani, G., Guerriero, F., Laporte, G., Musmanno, R.: Tabu search heuristics for the arc routing problem with intermediate facilities under capacity and length restrictions. Journal of Mathematical Modelling and Algorithms 3(3), 209–223 (2004)
[24] Ghiani, G., Improta, G., Laporte, G.: The capacitated arc routing problem with intermediate facilities. Networks 37(3), 134–143 (2001)
[25] Golden, B., DeArmon, J., Baker, E.: Computational experiments with algorithms for a class of routing problems. Computer and Operations Research 10(1), 47–59 (1983)
[26] Golden, B., Wong, R.: Capacitated arc routing problems. Networks 11(3), 305–316 (1981)
[27] Gunawan, S., Papalambros, P.: A Bayesian Approach to Reliability-Based Optimization With Incomplete Information. Journal of Mechanical Design 128(4), 909–918 (2006)
[28] Handa, H., Chapman, L., Yao, X.: Robust route optimization for gritting/salting trucks: a CERCIA experience. IEEE Computational Intelligence Magazine 1(1), 6–9 (2006)
[29] Handa, H., Chapman, L., Yao, X.: Robust Salting Route Optimization Using Evolutionary Algorithms. In: Yang, S., Ong, Y.S., Jin, Y. (eds.) Evolutionary Computation in Dynamic and Uncertain Environments. SCI, vol. 51, pp. 497–517. Springer, Heidelberg (2007)
[30] Herrmann, J.: A genetic algorithm for minimax optimization problems. In: Proceedings of the 1999 Congress on Evolutionary Computation, vol. 2, pp. 1099–1103. Citeseer (1999)

[31] Jaillet, P.: Probabilistic traveling salesman problems. Ph.D. thesis, Massachusetts Institute of Technology, Department of Civil Engineering (1985)
[32] Kalsi, M., Hacker, K., Lewis, K.: A comprehensive robust design approach for decision trade-offs in complex systems design. Journal of Mechanical Design 123(1), 1–10 (2001)
[33] Lacomme, P., Prins, C., Ramdane-Cherif, W.: Evolutionary algorithms for periodic arc routing problems. European Journal of Operational Research 165(2), 535–553 (2005)
[34] Laporte, G., Musmanno, R., Vocaturo, F.: An Adaptive Large Neighbourhood Search Heuristic for the Capacitated Arc-Routing Problem with Stochastic Demands. Transportation Science 160(1), 139–153 (2009)
[35] Lewis, A.: Robust regularization. Tech. rep., Simon Fraser University, Vancouver, Canada (2002)
[36] Li, M., Azarm, S., Boyars, A.: A new deterministic approach using sensitivity region measures for multi-objective robust and feasibility robust design optimization. Journal of Mechanical Design 128(4), 874–883 (2006)
[37] McIlhagga, M., Husbands, P., Ives, R.: A comparison of search techniques on a wing-box optimisation problem. In: Ebeling, W., Rechenberg, I., Voigt, H.-M., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 614–623. Springer, Heidelberg (1996)
[38] Mei, Y., Tang, K., Yao, X.: A Global Repair Operator for Capacitated Arc Routing Problem. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 39(3), 723–734 (2009)
[39] Mourão, M., Amado, L.: Heuristic method for a mixed capacitated arc routing problem: A refuse collection application. European Journal of Operational Research 160(1), 139–153 (2005)
[40] Papadrakakis, M., Lagaros, N., Plevris, V.: Design optimization of steel structures considering uncertainties. Engineering Structures 27(9), 1408–1418 (2005)
[41] Polacek, M., Doerner, K., Hartl, R., Maniezzo, V.: A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. Journal of Heuristics 14(5), 405–423 (2008)
[42] Tagmouti, M., Gendreau, M., Potvin, J.: Arc routing problems with time-dependent service costs. European Journal of Operational Research 181(1), 30–39 (2007)
[43] Taguchi, G.: Introduction to quality engineering. Asian Productivity Organization, Tokyo (1990)
[44] Tang, K., Mei, Y., Yao, X.: Memetic Algorithm with Extended Neighborhood Search for Capacitated Arc Routing Problems. IEEE Transactions on Evolutionary Computation 13(5), 1151–1166 (2009)
[45] Tillman, F.: The multiple terminal delivery problem with probabilistic demands. Transportation Science 3(3), 192–204 (1969)
[46] Ulusoy, G.: The fleet size and mix problem for capacitated arc routing. European Journal of Operational Research 22(3), 329–337 (1985)
[47] Waters, C.: Vehicle-scheduling problems with uncertainty and omitted customers. The Journal of the Operational Research Society 40(12), 1099–1108 (1989)
[48] Weise, T., Podlich, A., Gorldt, C.: Solving Real-World Vehicle Routing Problems with Evolutionary Algorithms. In: Chiong, R., Dhakal, S. (eds.) Natural Intelligence for Scheduling, Planning and Packing Problems. SCI, vol. 250, pp. 29–53. Springer, Heidelberg (2009)
[49] Weise, T., Podlich, A., Reinhard, K., Gorldt, C., Geihs, K.: Evolutionary Freight Transportation Planning. In: Giacobini, M., et al. (eds.) EvoWorkshops 2009. LNCS, vol. 5484, pp. 768–777. Springer, Heidelberg (2009)
[50] Wiesmann, D., Hammel, U., Back, T.: Robust design of multilayer optical coatings by means of evolutionary algorithms. IEEE Trans. Evol. Comput. 2(4), 162–167 (1998)