WCCI 2010 IEEE World Congress on Computational Intelligence
July, 18-23, 2010 - CCIB, Barcelona, Spain CEC IEEE

Capacitated Arc Routing Problem in Uncertain Environments

Yi Mei, *Student Member, IEEE*, Ke Tang, *Member, IEEE*, and Xin Yao, *Fellow, IEEE*

***Abstract*— In this paper, the Uncertain CARP (UCARP) is investigated. In UCARP, the demands of tasks and the deadheading costs of edges are stochastic and one has to design a robust solution for all possible environments. A problem model and a robustness measure for solutions are defined according to the requirements in reality. Three benchmark sets with uncertain parameters are generated by extending existing benchmark sets for static cases. In order to explore the solution space of UCARP, the most competitive algorithms for static CARP are tested on one of the generated uncertain benchmark sets. The experimental results showed that the optimal solution in terms of robustness in uncertain environment may be far away from the optimal one in terms of quality in a static environment and thus, utilizing only the expected value of the random variables can hardly lead to robust solutions.**

I. Introduction

As a classic combinatorial optimization problem with wide real-world applications, the Capacitated Arc Routing Problem (CARP) [1] in static environments has received much research interest. However, this is quite far away from the reality, where the problem parameters are often stochastic. For example, in snow removal problems, the amount of snow that has to be removed from the streets is not known exactly until the vehicles have reached them. In this paper, the corresponding optimization problem, called the uncertain CARP (UCARP), is investigated.

Unfortunately, although UCARP is much closer to the real-world applications, it has been overlooked so far. There has been only one uncertain version of CARP considered in previous work, i.e., the stochastic CARP (SCARP) proposed by Fleury et al. [2]. In SCARP, the demands of the tasks are considered as random variables. Due to the stochastic character of the demands, the feasibility of solution cannot be guaranteed and the total demand taken by a single route may exceed the capacity. The goal of SCARP is to find a solution and a recourse operation to minimize the total cost of the final feasible solution obtained by applying the recourse operation to the solution. The SCARP is a simple version of UCARP as only the demands of the tasks are stochastic and other uncertain parameters, such as the deadheading costs of the edges, are fixed in the problem.

More research has been carried out on uncertain versions of the node routing counterpart of CARP, i.e., the Vehicle Routing Problem (VRP), or even its one-vehicle special case called the Traveling Salesman Problem (TSP). In stochastic TSPs, the random variables taken into account include the presence of customers and the travel times between the customers. The TSP with stochastic customers was proposed by Jaillet [3] 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 demands [4], [5], [6] and on the stochastic presence of customers [5], [7], or both [8]. [9] gives a comprehensive survey of the aforementioned problems. [10] and [11] consider dynamic replanning for VRP in case of unforseen situations such as traffic jams.

In summary, there have been three stochastic factors considered in previous research work: (1) the presence of tasks (customers in VRP and edges in CARP); (2) the demands of the tasks and (3) the deadheading costs between the tasks. In fact, there should be a fourth stochastic factor: the availability of a path between each pair of vertices. For example, when a street is on its maintenance, it becomes unavailable to be traversed and thus disappears from the graph temporarily. These 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 demands of the tasks are combined in [8]). Here, we consider all the four stochastic factors in the problem.

An optimization problem like CARP can be characterized by the following four aspects: inputs, outputs, constraints and objectives. UCARP is different from CARP with respect to all the above four aspects. In this paper, we give the formal definition of UCARP. First, we point out the differences between UCARP and CARP with respect to the inputs, outputs and constraints, and provide a mathematical statement of UCARP. Second, in UCARP, it is no longer appropriate to minimize the total cost in a specific environment. Instead, one should find a solution that can show relatively good performance under all possible environments. In other words, the objective of UCARP is to find a robust solution. Based on such consideration, a performance measure of UCARP is defined in order to evaluate the robustness of a solution. After that, three sets of UCARP benchmark problems named the *Ugdb*, *Uval* and *Uegl* sets are generated by extending the corresponding *gdb* [12], *val* [13] and *egl* [14] sets for the static CARP, respectively. Finally, in order to investigate

The authors are with the Nature Inspired Computation and Applications Laboratory, School of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China. Xin Yao is also with CERCIA, the School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K. (emails: meiyi@mail.ustc.edu.cn, ketang@ustc.edu.cn, x.yao@cs.bham.ac.uk).
Corresponding author: Ke Tang (Phone: +86–551–3600547).

the fitness landscape of UCARP, two existing competitive algorithms for static CARP are tested on the simplest of the three extended UCARP benchmark set, i.e., the *Ugdb* set. The experimental results showed that it is difficult to find highly robust solutions if only the expected values of the random variables are utilized.

The rest of the paper is organized as follows: First, Section II gives the formal definition of UCARP. Then, based on practical requirements, a robustness measure for UCARP solutions is defined in Section III. In Section IV, three benchmark sets are generated by extending the corresponding three static benchmark sets. Afterwards, in Section V, two static CARP approaches are applied to UCARP instances to conduct a rough investigation and analysis of the fitness landscape of UCARP. Finally, conclusion and future work are presented in Section VI.

II. Problem Definition

Solving a CARP means to determine a routing plan on a given graph to serve a given set of edges and arcs (also called *tasks*) of the graph at a minimal total cost subject to the following constraints:

- Each route starts and ends at the *depot* vertex;
- Each task is served exactly by one route;
- The total demand of the tasks served by each route cannot exceed its capacity.

The total cost of the routing plan includes the cost induced by serving the tasks (also called the *serving cost*) and the cost induced by traversing the paths between the tasks (also called the *deadheading cost*).

Given a graph $G(V, E, A)$, where V, E and A represent the sets of vertices, edges and arcs, respectively. The deadheading cost matrix is denoted as $(dc_{ij})_{|V|\times|V|}$, where $dc_{ij} = dc(v_i, v_j)$ indicates the deadheading cost for traversing from vertex v_i to vertex v_j. Obviously, $dc_{ij} = \infty \Longleftrightarrow$there is no edge or arc from v_i to v_j. All edges $e \in E$ and arcs $a \in A$ are associated with a nonnegative demand $d(e)$ and $d(a)$, and a nonnegative serving cost $sc(e)$ and $sc(a)$. The edges and arcs with positive demands are called the tasks. There are m vehicles with capacity of Q are located at the depot $v_0 \in V$ to serve the task set $T = \{t \in E \cup A | d(t) > 0\}$.

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 $X_k = (x_{k1}, ..., x_{kl_k})$ is a sequence of vertices starting and ending at v_0, i.e., $x_{k1} = x_{kl_k} = v_0$, while 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, $y_{ki} = 0$. For the sake of convenience, the above solution representation scheme is called the vertex representation. Vertex representation is the most explicit representation scheme for CARP solutions since under such representation scheme, a CARP solution can be implemented without any transformation. Concretely, a solution can be implemented in the following way: For each vertex sequence X_k, the assigned vehicle starts from v_0 and traverse towards the next vertex in the sequence. If the corresponding 0-1 indicator in Y_k is 1, then the vehicle serves the path to be traversed. Otherwise, it traverses the path without serving it. The above operations are repeated until the vehicle finishes the sequence and returns to v_0. In addition to the vertex representation scheme, there are other representation schemes for CARP solutions proposed, such as the explicit task representation scheme [15] and the implicit task representation scheme [16]. When using the explicit task representation scheme, a solution is represented by a set of task sequences $\{R_1, ..., R_m\}$. During the implementation phase, each task sequence is transformed to a vertex sequence by connecting the adjacent tasks with the shortest path between them in the graph. Therefore, the explicit task representation is equivalent to the vertex representation during implementation. Similarly, the implicit task representation is equivalent to the vertex representation as well since it will be transformed to the explicit task representation first during implementation. In summary, although there are various solution representation schemes proposed to facilitate the search, they have to be transformed to the most explicit vertex representation during implementation.

In a static environment, CARP can be stated as follows:

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

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

$$\sum_{k=1}^{m}\sum_{i=1}^{l_k-1} y_{ki} = |T| \tag{3}$$

$$(x_{ki}, x_{k(i+1)}) \in E_R \cup A_R,\ \ \forall y_{ki} = 1 \tag{4}$$

$$(x_{k_1 i_1}, x_{k_1(i_1+1)}) \neq (x_{k_2 i_2}, x_{k_2(i_2+1)}),\quad \forall y_{k_1 i_1} = 1, y_{k_2 i_2} = 1, (k_1, i_1) \neq (k_2, i_2) \tag{5}$$

$$(x_{k_1 i_1}, x_{k_1(i_1+1)}) \neq (x_{k_2(i_2+1)}, x_{k_2 i_2}),\quad \forall y_{k_1 i_1} = 1, y_{k_2 i_2} = 1, (k_1, i_1) \neq (k_2, i_2) \tag{6}$$

$$\sum_{i=1}^{l_k-1} d(x_{ki}, x_{k(i+1)}) \times y_{ki} \leqslant Q,\ \ \forall k = 1, 2, ..., m \tag{7}$$

$$x_{ki} \in V,\ \ dc(x_{ki}, x_{k(i+1)}) < \infty,\ \ y_{ki} = 0\ \ or\ \ 1 \tag{8}$$

in constraints (5) and (6), the inequality $(k_1, i_1) \neq (k_2, i_2)$ means that at least one of the two inequalities $k_1 \neq k_2$ and $i_1 \neq i_2$ is satisfied. Objective (1) is to minimize the total cost $tc(S)$. Constraint (2) indicates all the routes start and end at the depot v_0. Constraints (3)–(6) guarantee that all the tasks are served exactly once. Constraint (7) is the capacity constraint, i.e., the total demands served by each route cannot exceed the capacity Q. Constraint (8) defines the domain of the variables.

Contrary to static CARP, in UCARP, the deadheading cost matrix and the task demands are random variables depending on the environmental parameter ξ. In practice, ξ

is affected by various real-world factors. For example, the surface temperature of a street in Salt Routing Optimization determines the demand of the street, while the traffic situation influences the deadheading cost matrix. For the sake of convenience, all the real-world factors are combined into the single environmental parameter ξ. Thereby, in UCARP, the deadheading cost matrix and the task demands can be denoted as functions of ξ, i.e., $(dc_{ij}(\xi))_{|V|\times|V|}$ and $d(t, \xi)$.

As mentioned in Section I, the stochastic factors in UCARP include (1) presence of tasks; (2) demands of tasks; (3) presence of paths between vertices and (4) deadheading costs between vertices. Specifically, in UCARP,

- each task (v_i, v_j) is present with probability p_{ij}, and absent with probability $1 - p_{ij}$. If it is present, its demand obeys a certain distribution, i.e., $P(d(v_i, v_j) \leqslant x) = \int_{-\infty}^{x} d(v_i, v_j, \xi_t(i, j)) \times pdf(\xi_{ij}^d)d\xi_{ij}^d$, where ξ_{ij}^d is the component of ξ affecting the demand of the task (v_i, v_j), and $pdf(\xi_{ij}^d)$ is the probability density function of ξ_{ij}^d. Otherwise, its demand is zero.
- each path (v_i, v_j) is present with probability q_{ij}, and absent with probability $1-q_{ij}$. If it is present, its deadheading cost obeys a certain distribution, i.e., $P(dc_{ij} \leqslant x) = \int_{-\infty}^{x} dc_{ij}(\xi_{ij}) \times pdf(\xi_{ij}^c)d\xi_{ij}$, where ξ_{ij}^c is the component of ξ affecting the deadheading cost of the path (v_i, v_j), and $pdf(\xi_{ij}^c)$ is the probability density function of ξ_{ij}^c. Otherwise, its deadheading cost is infinity, i.e., $dc_{ij} = \infty$.

In an uncertain environment, the problem objective is no longer to find a single global optimal solution, but to find a solution with the best expected quality under all possible environments, i.e., the most robust solution. For this reason, UCARP can be stated as follows:

$$\min\ R(S) \tag{9}$$

$$s.t.:\ x_{k1} = x_{kl_k} = v_0,\ \forall k = 1, 2, ..., m \tag{10}$$

$$\sum_{k=1}^{m}\sum_{i=1}^{l_k-1} y_{ki} = |T| \tag{11}$$

$$(x_{ki}, x_{k(i+1)}) \in E_R \cup A_R,\ \forall y_{ki} = 1 \tag{12}$$

$$\begin{aligned}&(x_{k_1 i_1}, x_{k_1(i_1+1)}) \neq (x_{k_2 i_2}, x_{k_2(i_2+1)}),\\ &\forall y_{k_1 i_1} = 1, y_{k_2 i_2} = 1, (k_1, i_1) \neq (k_2, i_2)\end{aligned} \tag{13}$$

$$\begin{aligned}&(x_{k_1 i_1}, x_{k_1(i_1+1)}) \neq (x_{k_2(i_2+1)}, x_{k_2 i_2}),\\ &\forall y_{k_1 i_1} = 1, y_{k_2 i_2} = 1, (k_1, i_1) \neq (k_2, i_2)\end{aligned} \tag{14}$$

$$E[\sum_{i=1}^{l_k-1} d(x_{ki}, x_{k(i+1)}, \xi) \times y_{ki}|\xi] \leqslant Q,\ \forall k = 1, 2, ..., m \tag{15}$$

$$x_{ki} \in V,\ dc(x_{ki}, x_{k(i+1)}, \xi) < \infty,\ y_{ki} = 0\ or\ 1 \tag{16}$$

The constraints (10)–(14) and (16) are exactly the same as those of the static counterpart (2)–(6) and (8). The differences between UCARP and static CARP lie in the following aspects: First, as shown in (9), the objective of UCARP becomes the robustness measure $R(S)$, whose definition will be given in Section III. Second, due to the stochastic character of task demands, the capacity constraint (7) cannot be guaranteed in UCARP. Therefore, the capacity constraint in UCARP is modified in such a way that the expected total demands (with respect to the environmental parameter ξ) served by each route does not exceed the capacity Q. The modified capacity constraint can be seen in (15).

III. Robustness Optimization in Uncertain Capacitated Arc Routing Problem

The concept of robust optimization in uncertain environment has been considered in the fields of operations research, engineering design and other scientific areas. So far, there have been various robustness measures proposed. Next, we will introduce several commonly used robustness measures.

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

$$R_w(x) = \sup_{x' \in \mathcal{N}(x)} f(x') \tag{17}$$

where $\mathcal{N}(x)$ stands for a predefined neighborhood of x.

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:

$$R_e(x) = E[f(x;\xi)|\xi] = \int f(x;\xi)d\xi \tag{18}$$

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.,

$$R_t(x) = Pr[f(x) \leqslant q] \tag{19}$$

4. Reliability-based Robustness Measure: In uncertain environments, the stochastic factors may appear not only in the objective functions, but also in the constraints. For such kind of problems, the practical feasibility of solution can no longer be guaranteed. Given the following optimization problem:

$$\min\ f(x) \tag{20}$$

$$s.t.\ \ g_i(x) \leqslant 0,\ i = 1, ..., I \tag{21}$$

$$h_j(x) = 0,\ j = 1, ..., J \tag{22}$$

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. An example of the measure is given in [17].

5. Repair-based Robustness Measure: Like the reliability-based robustness measure, this measure is used to deal with the stochastic factors appearing in constraints as well. It 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 be made feasible, i.e., $x \leftarrow \Phi(x, \xi)$. 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. [2] adopted the measure in the context of the CARP with stochastic task demands.

In summary, measures 1–3 tackle the stochastic factors in objective functions, while measures 4 and 5 deal with the stochastic constraints. in UCARP, the random variables include the deadheading costs $dc(v_i, v_j)$ and the task demands $d(t)$. From Eq. (1)–(8), we can see that $dc(v_i, v_j)$ will affect the objective function and the domain of $(x_{ki}, x_{k(i+1)})$ in constraint (8), while $d(t)$ will determine the satisfactory of constraint (7). Therefore, we need to pick a measure from 1–3 to handle the stochastic objective function, and pick one from 4 and 5 to handle the stochastic constraints. Here, measures 2 and 5 are picked. Measures 1–3 reflect different aspects of the objective function and can be chosen according to practical consideration. Here, we select the representative expectation performance to evaluate the stochastic total cost. In contrast, the reliability-based and repair-based robustness measures are quite different in tackling stochastic constraints. The reason why we select the repair-based robustness measure can be explained as follows: in UCARP, the stochastic constraints are hard constraints, and the infeasible solutions cannot be implemented. First, it is obvious that each vehicle cannot serve total demands larger than its capacity. Second, when the path between adjacent vertices becomes absent, the latter vertex can no longer be reached unless another path is found. Therefore, once a solution becomes infeasible due to the stochastic factors, it has to be made feasible by certain repair operator, or it will fail to be implemented. If using the reliability-based robustness measure, the system will collapse with probability $1 - P_0$. For this reason, the repair-based robust measure is more appropriate than the reliability-based robust measure for UCARP.

Based on the introduction of the repair-based robustness measure, a repair operator Φ has to be defined. As mentioned above, in UCARP, the feasibility of solution can be changed due to the stochastic character of task demands and presence of paths. 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 consideration, 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 repair operation should consider minimizing the modification of the solution.

Here, we assume the practical value of the random variables can only be known during the implementation. 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 $x_{k(i+1)}$. Based on the above assumption, the repair operation of Φ can be defined as follows: Given a solution $S = (\{X_1, ..., X_m\}, \{Y_1, ..., Y_m\})$ represented by the vertex representation scheme, 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} through the original path. Otherwise, the vehicle returns to the depot through the shortest path and update the capacity, then goes to x_{kj} through the shortest path 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} so as to finish the service of $(x_{kj}, x_{k(j+1)})$. After the above operations, the satisfactory of the capacity constraint can be guaranteed. On the other hand, for the infeasibility caused by the absence of the path between the adjacent vertices x_{ki} and $x_{k(i+1)}$, 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 [18]. 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. Algorithm 1 and 2 give the pseudo codes of the repair operators Φ_d and Φ_c, respectively.

In Algorithm 1 and 2, the function $X.pushback(a)$ 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. $\mathbf{0}_l$ 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 under 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) $S^{\xi} = \Phi_c(S')$. Then, the expected performance of a solution S is defined as the expected total cost of all possible S^{ξ}'s, i.e., the objective function (9) is defined as:

$$R(S) = E[tc(S^{\xi})|\xi] \tag{23}$$

```
Input: $S = (\{X_1, ..., X_m\}, \{Y_1, ..., Y_m\})$, $\xi$
Output: A solution satisfying capacity constraint
    $S' = \Phi_d(S, \xi)$
 1: for $k = 1 \rightarrow m$ do
 2:   Set $X'_k = (v_0)$, $Y'_k = ()$, $\Delta Q = Q$;
 3:   for $i = 1 \rightarrow |X_k| - 1$ do
 4:     if $y_{ki} = 0$ then
 5:       $X'_k.pushback(x_{k(i+1)})$, $Y'_k.pushback(0)$;
 6:     else
 7:       if $d(x_{ki}, x_{k(i+1)}, \xi) = 0$ then
 8:         $X'_k.pushback(x_{k(i+1)})$;
 9:         $Y'_k.pushback(0)$;
10:       else if $\Delta Q < d(x_{ki}, x_{k(i+1)}, \xi)$ then
11:         $X'_k.pushback(ESP(x_{k(i+1)}, v_0))$;
12:         $X'_k.pushback(v_0)$;
13:         $Y'_k.pushback(1)$;
14:         $Y'_k.pushback(\mathbf{0}_{|ESP(x_{k(i+1)}, v_0)|})$;
15:         $X'_k.pushback(ESP(v_0, x_{ki}))$;
16:         $Y'_k.pushback(\mathbf{0}_{|ESP(v_0, x_{ki})|})$;
17:         $X'_k.pushback(x_{ki})$, $Y_k.pushback(0)$;
18:         $\Delta Q \leftarrow \Delta Q + Q - d(x_{ki}, x_{k(i+1)}, \xi)$;
19:       else
20:         $X'_k.pushback(x_{k(i+1)})$, $Y'_k.pushback(1)$,
            $\Delta Q \leftarrow \Delta Q - d(x_{ki}, x_{k(i+1)}, \xi)$;
21:       end if
22:       for $j = i + 1 \rightarrow |X_k| - 1$ do
23:         if $Y_{kj} = 1$ then
24:           break;
25:         end if
26:       end for
27:       if $Y_{kj} = 1$ and
          $\Delta Q < E[d(x_{kj}, x_{k(j+1)}, \xi)|\xi]$ then
28:         $X'_k.pushback(ESP(x_{k(i+1)}, v_0))$;
29:         $X'_k.pushback(v_0)$;
30:         $Y'_k.pushback(\mathbf{0}_{|ESP(x_{k(i+1)}, v_0)|+1}$;
31:         $X'_k.pushback(ESP(v_0, x_{kj}))$;
32:         $X'_k.pushback(x_{kj})$;
33:         $Y'_k.pushback(\mathbf{0}_{|ESP(v_0, x_{kj})|+1})$;
34:         $i \leftarrow j$, $\Delta Q \leftarrow Q$;
35:       end if
36:     end if
37:   end for
38: end for
39: return $S' = (\{X'_1, ..., X'_m\}, \{Y'_1, ..., Y'_m\})$;
```

Algorithm 1: The pseudo code of Φ_d

IV. Benchmark Generation

Since there currently exists no benchmark instance for UCARP, we decided to generate them by ourselves. Intuitively, we can extend the existing static CARP benchmark instances to UCARP instances. In [2], the well-known *gdb* test set with static CARP instances was extended to a stochastic CARP test set by replacing the deterministic demand $d(t_i)$

```
Input: A solution obtained by $\Phi_d$:
    $S' = (\{X'_1, ..., X'_m\}, \{Y'_1, ..., Y'_m\})$, $\xi$
Output: A feasible solution $S^{\xi} = \Phi(S, \xi)$
 1: for $k = 1 \rightarrow m$ do
 2:   Set $X^{\xi}_k = (v_0)$, $Y^{\xi}_k = ()$;
 3:   for $i = 1 \rightarrow |X'_k| - 1$ do
 4:     if $dc(x'_{ki}, x'_{k(i+1)}, \xi) < \infty$ then
 5:       $X^{\xi}_k.pushback(x'_{k(i+1)})$, $Y^{\xi}_k.pushback(y'_{ki})$;
 6:     else
 7:       $X^{\xi}_k.pushback(SP(x'_{ki}, x'_{k(i+1)}, \xi))$,
          $X^{\xi}_k.pushback(x'_{k(i+1)})$;
 8:       $Y^{\xi}_k.pushback(\mathbf{0}_{|SP(x'_{ki}, x'_{k(i+1)}, \xi)|+1})$;
 9:     end if
10:   end for
11: end for
12: return $S^{\xi} = (\{X^{\xi}_1, ..., X^{\xi}_m\}, \{Y^{\xi}_1, ..., Y^{\xi}_m\})$;
```

Algorithm 2: The pseudo code of Φ_c

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 can generate our benchmark instances similarly, but the Gamma distribution is chosen for the random variables. This is because their values must be nonnegative, and the Gamma distribution is one of the commonly used distributions having nonnegative supports. 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 UCARP, the random variables should satisfy the following distributions:

$$D(v_i, v_j) \begin{cases} \sim G(k^d_{ij}, \theta^d_{ij}), & r < p_{ij}; \\ = 0, & otherwise. \end{cases}$$

$$DC(v_i, v_j) \begin{cases} \sim G(k^c_{ij}, \theta^c_{ij}), & r < q_{ij}; \\ = \infty, & otherwise. \end{cases}$$

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 Gaussian distribution when the shape parameter k becomes infinite. According to the idea of UCARP instance generation in [2], the random variables in UCARP should have the following properties: Property 1:

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

Property 1 can be realized by setting a sufficiently large k. Here, we set $k = 20$ for all the random variables. The probability density functions of the Gamma distribution with the shape parameter $k = 20$ and various θ values are illustrated in Fig. 1.

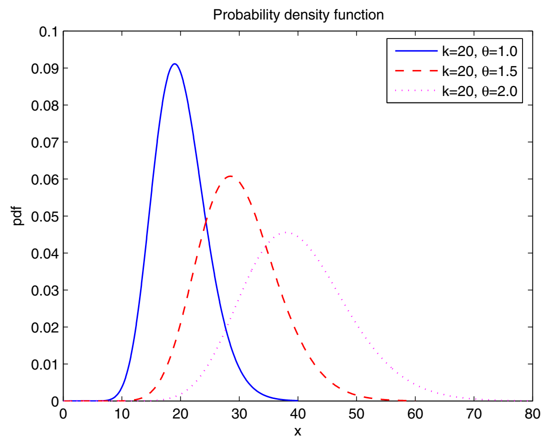

Fig. 1. Probability density function of Gamma distribution with $k = 20$ and various θ values

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

$$E[D(v_i, v_j)] = p_{ij}k_{ij}^d\theta_{ij}^d + (1 - p_{ij}) \times 0 = d(v_i, v_j) \quad (24)$$

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

$$E[DC(v_i, v_j)] = k_{ij}^c\theta_{ij}^c = dc(v_i, v_j) \quad (25)$$

From Eq. (24) and (25), we can get $\theta_{ij}^d = \frac{d(v_i,v_j)}{p_{ij}k_{ij}^d}$ and $\theta_{ij}^c = \frac{dc(v_i,v_j)}{k_{ij}^c}$. Besides, 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 UCARP instances can be downloaded on http://home.ustc.edu.cn/∼meiyi, along with the extended instances for the *gdb*, *val*, and *egl* sets, namely the *Ugdb*, *Uval* and *Uegl* sets, respectively. For each static instance, 30 uncertain instances are generated one by one by the instance generator with the starting random seed of 0. If necessary, users can also generate more uncertain instances with other random seeds.

V. Experimental Studies

The stochastic character of random variables makes the fitness landscape of UCARP much more complicated than CARP. To investigate the fitness landscape of UCARP and the impact of the stochastic factors on the performance of the search algorithms, the two competitive static CARP optimization approaches, RTS* [19] and MAENS [15], were applied to the *Ugdb* set, which is the simplest and smallest test set among the three UCARP benchmark sets generated in Section IV. In this way, it is easier to observe how the performance of the algorithms are influenced by the uncertain environment, but not the complicatedness of the problem itself. As demonstrated in [19] and [15], the two selected algorithms are able to reach the global optima for all the static version of the *Ugdb* 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 *Ugdb* instances are exactly the corresponding static values of the *gdb* instances. Thus, by solving the *gdb* instances, the algorithms can be seen as solving the *Ugdb* 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 performance defined in Section III ($R(S_{ij})$) is calculated in terms of the average total cost of the 30 corresponding *Ugdb* instance samples generated in Section IV, i.e.,

$$R(S_{ij}) = \frac{1}{30}\sum_{k=1}^{30} tc(S_{ij,\xi_k}) \quad (26)$$

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 I presents the experimental results. In the table, the columns headed "Best $tc(S)$" and "Best $R(S)$" stand for the solution with the lowest $tc(S)$ and that with the lowest $R(S)$ among all the recorded solutions. The column headed "Best Known $R(S)$" 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. (1) and $R(S)$ is defined in Eq. (6). 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 I, 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 *Ugdb*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, it can be concluded that for a static CARP instance, there often exist multiple globally optimal solutions. However, their robustness in the corresponding uncertain 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 UCARP instance may be quite far away from the global optimum in its static counterpart. Comparing with the results obtained

TABLE I
THE EXPERIMENTAL RESULTS OF MAENS AND RTS* ON THE *Ugdb* SET. THE OPTIMAL $c^{tot}(S)$'S ARE MARKED IN BOLD.

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

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 *Ugdb* instances by applying algorithms to the *gdb* counterparts, it is difficult to achieve highly robust solutions.

In summary, the following observations can be drawn:

- The robustness of solution in UCARP is not proportional to its absolute performance in static CARP;
- Static CARP instances often have multiple global optimal solutions. However, their robustness in the corresponding uncertain version may be quite different;
- Only utilizing the expected information cannot lead to highly robust solutions.

VI. CONCLUSION AND FUTURE WORK

In this paper, we investigate a new variant of the well-known CARP, i.e., the Uncertain CARP (UCARP). In contrast to its static counterpart, in UCARP, the environment parameters are uncertain, and the objective is to find a solution that can keep a relatively good average performance over all the possible environments instead of having optimal performance for one specific static environment. In our study, the UCARP with random variables denoting the presence of tasks, the demands of the tasks, the availability of paths between vertices, and the traversal costs of the paths is formally defined. A performance measure to indicate solution quality for such problems is also defined on basis of all stochastic factors. Furthermore, three benchmark sets for UCARP are generated by extending the three commonly used static CARP benchmark sets. The two most competitive algorithms for static CARP, i.e., MAENS [15] and RTS* [19], were tested on the simplest generated benchmark set, i.e., the *Ugdb* set. It is found that, although the two algorithms showed excellent performance for static CARP, they were not able to find robust solutions for UCARP. Therefore, the future work is to design new algorithms that can find more robust solutions by taking advantage of more information random character. 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.

ACKNOWLEDGEMENT

This work is partially supported by a National Natural Science Foundation of China grant (No. U0835002), the Fund for Foreign Scholars in University Research and Teaching Programs (Grant No. B07033) and an EPSRC grant (EP/E058884/1) on "Evolutionary Algorithms for Dynamic Optimisation Problems: Design, Analysis and Applications."

REFERENCES

[1] M. Dror, *Arc routing. Theory, solutions and applications.* Boston: Kluwer Academic Publishers, 2000.
[2] G. Fleury, P. Lacomme, and C. Prins, "Evolutionary algorithms for stochastic arc routing problems," *Lecture Notes in Computer Science*, vol. 3005, pp. 501–512, 2004.
[3] P. Jaillet, "Probabilistic traveling salesman problems," Ph.D. dissertation, M. I. T., Dept. of Civil Engineering, 1985.
[4] F. Tillman, "The multiple terminal delivery problem with probabilistic demands," *Transportation Science*, vol. 3, no. 3, pp. 192–204, 1969.

[5] D. Bertsimas, "Probabilistic combinatorial optimization problems," Ph.D. dissertation, Massachusetts Institute of Technology, Dept. of Mathematics, 1988.

[6] M. Dror, G. Laporte, and P. Trudeau, "Vehicle routing with stochastic demands: Properties and solution frameworks," *Transportation Science*, vol. 23, no. 3, pp. 166–176, 1989.

[7] C. Waters, "Vehicle-scheduling problems with uncertainty and omitted customers," *The Journal of the Operational Research Society*, vol. 40, no. 12, pp. 1099–1108, 1989.

[8] D. Bertsimas, "A vehicle routing problem with stochastic demand," *Operations Research*, pp. 574–585, 1992.

[9] M. Gendreau, G. Laporte, and R. Séguin, "Stochastic vehicle routing," *European Journal of Operational Research*, vol. 88, no. 1, pp. 3–12, 1996.

[10] T. Weise, A. Podlich, K. Reinhard, C. Gorldt, and K. Geihs, "Evolutionary Freight Transportation Planning," *Proceedings of the EvoWorkshops 2009 on Applications of Evolutionary Computing: EvoCOMNET, EvoENVIRONMENT, EvoFIN, EvoGAMES, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, EvoNUM, EvoSTOC, EvoTRANSLOG*, pp. 768–777, 2009.

[11] T. Weise, A. Podlich, and C. Gorldt, "Solving Real-World Vehicle Routing Problems with Evolutionary Algorithms," *Natural Intelligence for Scheduling, Planning and Packing Problems*, pp. 29–53, 2009.

[12] J. DeArmon, "A comparison of heuristics for the capacitated Chinese postman problem," Master's thesis, University of Maryland, 1981.

[13] E. Benavent, V. Campos, A. Corberan, and E. Mota, "The capacitated arc routing problem: lower bounds," *NETWORKS-NEW YORK-*, vol. 22, pp. 669–669, 1992.

[14] R. Eglese, "Routeing winter gritting vehicles," *Discrete Applied Mathematics*, vol. 48, no. 3, pp. 231–244, 1994.

[15] K. Tang, Y. Mei, and X. Yao, "Memetic Algorithm With Extended Neighborhood Search for Capacitated Arc Routing Problems," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1151–1166, 2009.

[16] P. Lacomme, C. Prins, and W. Ramdane-Chérif, "Competitive memetic algorithms for arc routing problem," *Annals of Operational Research*, vol. 131, no. 1-4, pp. 159–185, 2004.

[17] K. Deb, S. Gupta, D. Daum, J. Branke, A. Mall, and D. Padmanabhan, "Reliability-Based Optimization Using Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1054–1074, 2009.

[18] E. W. Dijkstra, "A Note on Two Problems in Connection with Graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[19] Y. Mei, K. Tang, and X. Yao, "A Global Repair Operator for Capacitated Arc Routing Problem," *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 39, no. 3, pp. 723–734, 2009.