2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A Memetic Algorithm with a New Split Scheme for Solving Dynamic Capacitated Arc Routing Problems

Min Liu, Hemant Kumar Singh and Tapabrata Ray

***Abstract*— Capacitated arc routing problems (CARPs) are usually modeled as static problems, where all information about the problem is known in advance and assumed to remain constant during the course of optimization. However, in practice, many factors such as demand, road accessibility, vehicle availability etc. change during the course of a mission and the routes of each vehicle must be reconfigured dynamically. This problem is referred to as dynamic capacitated arc routing problem (DCARP). In this study, a memetic algorithm with a new split scheme for DCARPs is proposed. This algorithm is capable to solve DCARPs with variations in vehicle availability, road accessibility, added/canceled tasks or demands and traffic congestions. The algorithm is also capable of solving static CARPs. The performance of the algorithm is reported on a 10-node and three 100-node examples in order to demonstrate the efficacy of the algorithm in solving static and dynamic problems.**

I. Introduction

The capacitated arc routing problem (CARP) is a challenging combinatorial problem, wherein a set of required arcs, each with a fixed demand, are served by a fleet of homogeneous vehicles of finite capacity while attempting to keep the total distance traveled to a minimum. However, in most real-life applications (winter gritting, street sweeping, inspection of pipe distribution networks etc.), certain unexpected events invariably happen while the vehicles are en route. Examples of such unpredicted events include change in demand (i.e., new customer requests/cancellations), unexpected road blocks, traffic congestion and vehicle breakdowns. Under such situations, the originally planned vehicle routes need to be reconfigured, making it a *dynamic* CARP (DCARP).

In DCARP, part or all of the information (such as demands, road accessibility etc.) is unknown and revealed dynamically while the vehicles are in service. Limited studies have been reported on solving DCARPs. With the availability of advanced global positioning systems, the area is likely to gain greater attention in coming years. For vehicle routing in dynamic scenarios, DCVRP has been studied in [1], and classified into two types, *deterministic* and *stochastic*. Similar classification can be extended to DCARP, with the difference being that the former deals with serving nodes, while latter deals with serving edges. In the deterministic form, there is no historical data available, and vehicle routes are generated based on current requirements. In the stochastic form, useful historical data about possible tasks, vehicle availability, roads maintenance etc. is available in order to schedule the vehicle routes. After initial scheduling, in both these forms, the vehicle routes are redefined continuously based on directions from the central decision maker.

Min Liu, Hemant Kumar Singh and Tapabrata Ray are with the School of Engineering and Information Technology, University of New South Wales, Canberra, Australia (email: min.liu@student.adfa.edu.au, {h.singh,t.ray}@adfa.edu.au).

The limited studies reported in solving DCARP include solving (deterministic) winter gritting problem with time-dependent service costs (distance) [2] and solving dynamic salting route with dynamic demand affected by road surface temperature [3]. In [2], a variable neighborhood descent heuristic, initially developed for the static version of the problem (where all service cost functions are known in advance and do not change thereafter), is adapted to handle the dynamic variant. In [3], a new salting route optimization system was proposed which combines Evolutionary Computation (EC) with the neXt generation Road Weather Information Systems (XRWIS). XRWIS is a forecast system used for predicting road surface temperatures and conditions across the road network over a 24-hour period. ECs were used to optimize a series of salting routes for winter gritting by considering XRWIS temperature data along with treatment vehicle and road network constraints.

So far, the existing research has only focused on DCARP with variation of distances and demands of tasks. However, in real life, dynamically revealed information may consist of at least six factors, namely, vehicle availability, road accessibility, new added tasks, canceled tasks, variation of traffic conditions and variation of demands. In this study, dynamic problems containing these variations are discussed and a memetic algorithm with a new split scheme embedded with a path repair operator is proposed to solve such problems.

The remaining sections of this paper are organized as follows. In Section II, the problem statement is presented. The description of the proposed algorithm (MASDC) is given in Section III, and the numerical experiments are presented in Section IV. Conclusions are discussed in Section V.

II. Problem Statement

Capacitated arc routing problem (CARP) is a combinatorial optimization problem, in which the aim is to service a set of tasks with a fleet of homogeneous vehicles (without exceeding their capacities), while minimizing the total distance traveled. In DCARP studied here, six factors can change during the course of operation, which are vehicle availability, road accessibility, added/canceled tasks, road congestions and change in tasks' demands. In this study, the vehicle speed is considered constant and the distance of the road is manipulated to reflect the effect of congestion. If the vehicles are unavailable due to breakdowns, the control center needs to assign other vehicles to complete the remaining tasks.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

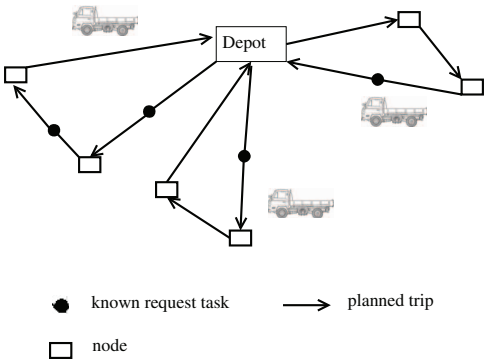
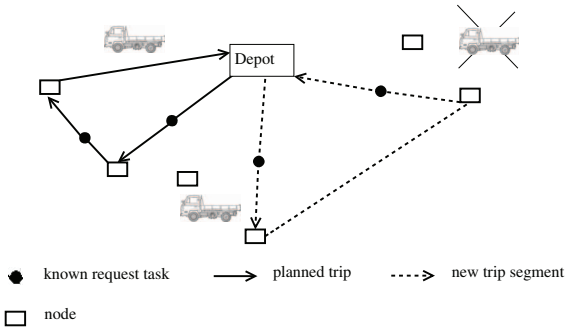


Fig. 1. A simple case of static CARP (I_0)

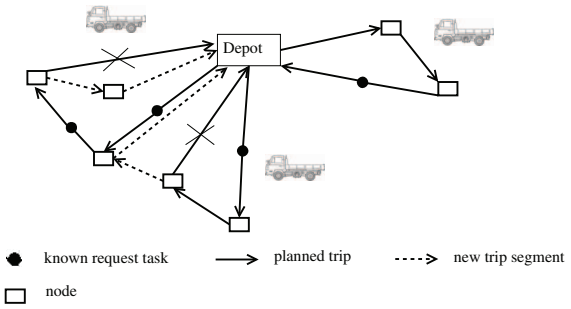
When certain roads are inaccessible (e.g. due to weather hazards) or there are congestions, the control center may need re-route the vehicles via different paths than originally planned. When some demands (tasks) are added or canceled, there is a need for rescheduling based on the position and availability of the vehicles. If certain demands of existing tasks are changed, the control center needs to reschedule the vehicles' routes to avoid exceeding of capacities, and at the same time make the total service distance as short as possible. Figure 1 shows a simple case for a static CARP (I_0) at the beginning, while Figure 2 illustrates the six factors of possible interruptions/changes considered in the DCARP model. The attributes of DCARP are listed below.

1) The fleet of vehicles may be homogeneous (same capacity) or heterogeneous (different capacities).
2) The vehicles may start from different depots.
3) Any vehicle may suffer a breakdown during its course of service.
4) A broken down vehicle is towed to its depot and repaired. The vehicle may be used subsequently for the mission and the distance from the site of breakdown to its depot is not accounted in the formulation. Furthermore, the repaired vehicle starts at its depot fully refilled.
5) Any road in the network may become inaccessible during the mission. Such inaccessibility could be short term (e.g. maintenance) or long term (e.g. land slides).
6) New tasks may be added in and some existing tasks may be canceled during the course of service.
7) Demand associated with any task may change over time (e.g., the amount of snow requiring removal usually grows with time). The demand can exceed vehicles' capacities for DCARP (in which case multiple trips are done to serve them).
8) The speed of each vehicle is assumed constant.
9) Information about the dynamic changes is conveyed to the control center without any time delay.
10) Each vehicle starts and ends at its own depot.
11) Each task is completed exactly once.

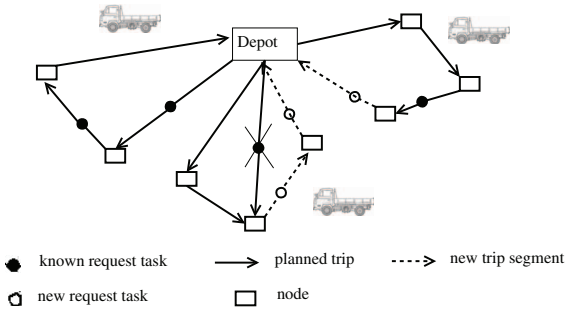
The objective of DCARP is to use available vehicles to serve all the required tasks (taking into account dynamic



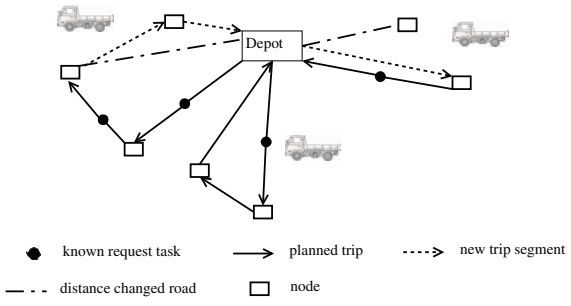
(a) Change in vehicle availability



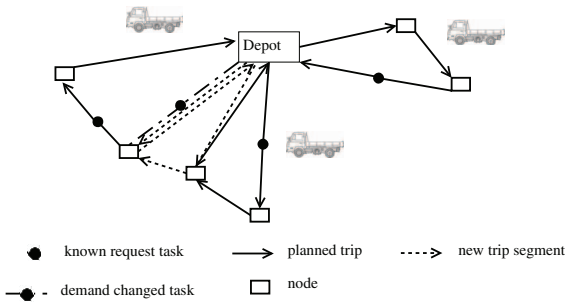
(b) Change in road accessibility



(c) Change in tasks, i.e., added and/or canceled tasks



(d) Change induced by congestion



(e) Change in demand of tasks

Fig. 2. Six types of possible interruptions/changes

changes) while traveling minimum total distance.

In DCARP, if the *interruptions/changes* occur at t_i ($i = 1, 2, ..., G_{max}$), then the original instance I_0 gets updated to generate new instances I_1, I_2,..., $I_{G_{max}}$ incorporating the changes.

III. Proposed Memetic Algorithm with new Split scheme for DCARP (MASDC)

In this section, a memetic algorithm is proposed to solve DCARPs. The algorithm incorporates a new split scheme with a path repair operator in order to handle the attributes of DCARP (compared to a static CARP). For solving a DCARP, an algorithm must be able to deal with the following situations (which do not occur in a standard static CARP problem).

1) Vehicles can have different depots/starting points. Handling this situation is necessary because when the instance changes, the vehicles may be in the process of serving existing tasks. Hence for the updated problem, the vehicles will not start from the original/common depot, but from their existing locations.
2) Vehicles have different capacities. When the change occurs in the problem, the vehicles may have served some of the tasks already, and hence will not be at their full capacities. Therefore algorithm must be able to route vehicles which may not necessarily have the same capacity.
3) The total demand may be larger than the sum of existing available vehicles' capacity. When/if the demands change, they total new demand may exceed the total capacities of the vehicles. Hence the algorithm must be able to have provision for dealing with this situation (e.g. by re-filling the vehicles).

MASDC is a memetic algorithm, combining features from global and local search, and has four key steps: split method, parent selection, crossover, and local search. A potential solution is represented using a sequence of tasks. From this sequence, to generate a vehicle schedule, a *split* method is used, which forms a solution. The total distance for this schedule gives the fitness of the individual (lower distance $\Rightarrow$ higher fitness). A population of individuals is maintained, and in each generation, two parents are selected from the population using Roulette Wheel [4] based selection to generate two offspring individuals via crossover operation. Thereafter, the local search is performed (with a given probability) from the the better offspring individual among the two as the starting point. The improved individual replaces the worst one in the existing population. The process continues until the maximum function evaluations are reached.

MASDC uses a conditional check to identify if the problem at a given time instant resembles static or dynamic CARP. If the problem at a given instant resembles static CARP (homogeneous vehicles, one depot and the total demand is less than the sum of existing available vehicles' capacity), then the methods used for split, parent selection, crossover, and local search are Ulusoy heuristic [5], Multiple Sequence Alignment (MSA) inspired selection, Hybrid recombination and Extended local search respectively as discussed in detail in [6]. The details of these operators are omitted here for brevity. However, if the problem resembles dynamic CARP (heterogeneous vehicles and/or different depots and/or total demand exceeds capacity), then the new proposed split scheme, Roulette wheel based selection, Random key crossover and Hill-climb local search method are used. These are described in more detail in the following subsections. The pseudo code of the MASDC is presented in Algorithm 1.

At any given time t_i ($i \epsilon \{0, 1, \ldots G_{max}\}$), the vehicles start serving the tasks obtained using the best solution from the search at time t_i. When the interruptions/change happen next at t_{i+1}, the vehicles stop at their current positions (nearest node) and the problem information (e.g. vehicle's left capacity, positions of vehicles and status of tasks etc), is updated.

Algorithm 1 Proposed memetic algorithm with new split scheme for DCARP (MASDC)

Require: Population size (P_s), Maximum number of function evaluations (MFE), r_{LS}, Times t_1, t_2,..., $t_{G_{max}}$ when interruptions/changes occur, instances I_0, I_1, I_2,..., $I_{G_{max}}$

for $i = 1$ to G_{max} **do**
  if $i = 1$ **then**
    Read instance I_0;
  else
    Read updated instance I_{i-1};
  end if
  Generate an initial population Q;
  while Number of solutions evaluated $<= MFE$ **do**
    if The graph is detected as basic (static) CARP **then**
      Apply MSA inspired selection operator to select P_s/2 pairs of parents;
      Apply hybrid recombination to generate a child population Q_C (size of P_s);
      Evaluate Q_C;
      Sort ($Q \cup Q_C$) and keep the best P_s individuals in Q;
      Select the best individual in Q as S_a;
      if *rand*[0,1] $\leq r_{LS}$ **then**
        Apply local search to S_a to generate S_c;
        Replace the worst individual in Q with S_c if it is better than it and not a clone of any individuals in Q;
      end if
      Sort the individuals in Q;
    else
      Apply Roulette Wheel selection operator to select two parents P_{c1} and P_{c2};
      Apply Random Keys crossover operator to P_{c1} and P_{c2} to generate a child S_a;
      if *rand*[0,1] $\leq r_{LS}$ **then**
        Apply local search to S_a to generate S_b;
        if $S_b \notin Q$ **then**
          $Q \leftarrow Q \cup S_b$
        else if $S_a \notin Q$ **then**
          $Q \leftarrow Q \cup S_a$;
        end if
      else if $S_a \notin Q$ **then**
        $Q \leftarrow Q \cup S_a$;
      end if
      Sort the chromosomes in Q and keep the best P_s solutions in Q;
    end if
  end while
end for

A. Solution Representation and Initialization

An individual (chromosome) of DCARP is represented as a list of tasks i.e. edge IDs. A set of chromosomes are generated by applying path-scanning (PS) heuristic [7] without considering the capacity constraints to form the initial population Q. The initialization phase terminates when P_s unique individuals have been generated. In the event P_s unique individuals cannot be generated within prescribed number of trials (=100), random individuals are added. The solutions are then generated from the chromosomes by applying *Distance based Split Scheme* (proposed in the next subsection) to split the individuals into an ordered list of tasks for each vehicle. The total distance D for this trip list defines the fitness of the individual.

B. Distance Based Split Scheme

A commonly used split method to evaluate a chromosome in existing CARP algorithms is Ulusoy heuristic [5]. Ulusoy heuristic is an exact algorithm that splits the chromosome in a way that minimizes total trip distance D. However, the method's applicability is restricted to basic CARP instances (homogeneous vehicles, same depot and total demand less than vehicle capacity). Furthermore, the heuristic does not take into account the number of available vehicles (only considers capacity), and hence the solution obtained from it may be infeasible (requires more vehicles than available). Since the above conditions are not valid for the DCARP instances, Ulusoy heuristic can not be used for splitting/evaluating the chromosomes. To handle these cases (heterogeneous vehicles, different depots, total demand more than vehicle capacity), a new *Distance based Split Scheme* embedded with a repair operator is proposed here. The details of the scheme are as follows:

1) Given a chromosome, randomly split it into N_V (number of vehicle) parts. For example, assume there are three vehicles V_1, V_2 and V_3 located at nodes 1, 2 and 3 respectively. A chromosome $S = \{2, 8, 6, 12, 1, 3, 5, 11, 7, 10, 4, 9\}$ is split into three parts, e.g. $S_1 = \{\{2, 8, 6\}$, $S_2 = \{12, 1, 3, 5, 11\}$, $S_3 = \{7, 10, 4, 9\}\}$.
2) Consider the first set of nodes, S_1. A vehicle which has overall least distance from the nodes of this set is sought to service this set. This is done as follows. Calculate the sum of distances of V_1 (using Dijkstra's algorithm [8]) from its current position to each task's start node and end node in S_1, e.g., Dp_{V_1} = sum ($distance$(node 1, start node of task 2) + $distance$(node 1, end node of task 2) + $distance$(node 1, start node of task 8) + $distance$(node 1, end node of task 8) + $distance$(node 1, start node of task 6) + $distance$(node 1, end node of task 6)). Similarly, calculate Dp_{V_2} and Dp_{V_3}. The vehicle which has the least distance among the three is assigned to service S_1.
3) Exclude the already assigned tasks from S and repeat this process using the remaining vehicles progressively until all tasks/vehicles are assigned.
4) Calculate the total distance of the three vehicles' routes as D_{R1}. During the process of serving, certain vehicle may encounter with a situation in which its remaining capacity is insufficient to serve the next task, making the solution infeasible. In such case, a *Path Repair Operator* is introduced and adopted here: if a vehicle's remaining capacity is not sufficient to serve its next task, then the vehicle is routed back to its depot to get refilled and then returns to the start node of the next task to continue its service. This does not affect the task list assigned to the vehicle, only the total distance required by the vehicle to finish the tasks. The distance from its current position to its depot and the distance from depot to the start node of next task are calculated by Dijkstras algorithm and are considered when calculating the solution's fitness.
5) Three trials of the process outlined above (Steps 1-4) are done (with different split locations) to calculate three possible configurations with distances D_{R1}, D_{R2} and D_{R3}. The configuration with the minimum distance is used to assign the fitness of the chromosome.

The pseudo code of proposed split scheme is presented in Algorithm 2.

Algorithm 2 Distance based Split Scheme

Require: A chromosome S, Number of vehicles N_V, Vehicle IDs V_1, V_2... V_{N_V}
1: $VSet \leftarrow V_1$, V_2... V_{N_V}
2: **for** $i = 1$ to 3 **do**
3: Randomly split S into N_V parts, S_1, S_2,...,S_{N_V}
4: **for** $j = 1$ to N_V **do**
5: **for** $k = 1$ to size($VSet$) **do**
6: Consider set S_j. Calculate the sum of distances of V_k from its current position to each task's start node and end node of the S_j and save the distance into Dp_{V_k}.
7: **end for**
8: The vehicle with shortest distance Dp is assigned to service S_j
9: Remove set S_j from $VSet$.
10: **end for**
11: Calculate the distance for this configuration as D_i. If the demand of a set assigned to a vehicle is higher than its capacity, use *Repair operator* in D_i calculation.
12: **end for**
13: Assign *min*($\{D_1, D_2, D_3\}$) as the distance D of chromosome S

C. Crossover

Commonly used crossovers for CARP, namely order crossover (OX) [9] and one-point/two-point crossover [10], are designed for circular permutations (like CARP tours). But the generated offspring might be infeasible with missed or repeated tasks. To counter this, some researchers have proposed repair algorithms [11] to recreate feasible trips. However, repair algorithms can consume a considerable amount of time, resulting in longer convergence time. Alternatively chromosomal representation like the one introduced by Bean [12] in which a random numbers encoding structure has been proposed, constituting so called Random Keys GA (RKGA). In our study, the random key crossover [12], [13] is adopted.

Random key operator works in the following way. First, two individuals P_{c1} and P_{c2} are identified using a roulette wheel based selection. Then, a set of Y (= total number of tasks) unique numbers, each number corresponding to a task in P_{c1} are sampled using uniform distribution between A and B (in our experiments A = 1, B = 4000). For the discussion assume them as {2100, 569, 3, 888, 970, 1100, 30}. The sorted list of random numbers is {3, 30, 569, 888, 970, 1100, 2100}. The original chromosome {6, 1, 5, 4, 2, 3, 7} is encoded such that the element 3 in the sorted list of random numbers is inserted in location 6, 30 in location 1, 569 in location 5 and so on resulting in an encoded chromosome as {30, 970, 1100, 888, 569, 3, 2100}. The process is repeated for P_{c2}, the encoded form of which is {90, 220, 457, 140, 1400, 700, 550}. Following two-point crossover [10] in the encoded space, the child chromosomes R_1* and R_2* assumes the form {30, 970, 1100, 140, 1400, 3, 2100} and {90, 220, 457, 888, 569, 700, 550}. A decoding of the chromosomes R_1* and R_2* back to the original space results in {6, 1, 4, 2, 3, 5, 7} and {1, 2, 3, 7, 5, 6, 4} respectively, where 6 denotes the position of the smallest random number in the encoded chromosome, 1 denotes the position of next smallest random number and so on. These offspring are guaranteed to be feasible and a random one is selected.

D. Local Search

In order to make the algorithm more efficient, local search is performed from the best individual S_a in the population Q with a probability r_{LS}. Local search performs successive phases that scan all pairs of tasks (Y_i, Y_j) and tries three moves [14]:

1) *Single insertion*: move task Y_i after task Y_j, or before Y_j if Y_j is the first task of its trip.
2) *Double insertion*: move adjacent tasks (Y_i, Y_{i+1}) after task Y_j, or before Y_j if Y_j is the first task of its trip.
3) *Swap*: swap tasks Y_i and Y_j.

One solution is generated using each of these moves, and the best individual (with the shortest distance) out of the three move operators is is selected as S_b. If this solution is better than the worst solution in the population, S_b replaces it. The pseudo code of the local search process is presented in Algorithm 3.

The best solution obtained using the search at the given instant t_i is used by the vehicles to start serving the tasks. When the next interruptions/changes happen at t_{i+1}, the vehicles stop at their current positions (nearest node) and the following information gets updated:

1) Currently available vehicle's information, i.e., vehicles' capacities and locations.
2) Updated Graph if certain roads become inaccessible.
3) The tasks yet to be served from previous time step, added and/or canceled tasks. If a vehicle has completed less than half of a task when the change occurred, this task is carried over to the new instance unserved, whereas if more than half of the task was complete, it is assumed to have been already served in the new instance.
4) Change in demands of existing tasks if applicable.

Algorithm 3 Local Search Algorithm

Require: Probability of invoking local search r_{LS}, Starting individual S_a
  //Single insertion
  for $i = 1$ to Number of tasks in S_a **do**
    for $j = i + 1$ to Number of tasks in S_a **do**
      Move the position of task Y_i to the position of task Y_j
      Use *split scheme* to calculate the this chromosome's fitness
    end for
  end for
  $S_a^1 \leftarrow$ The chromosome with the maximum fitness
  //Double insertion
  for $i = 1$ to Number of tasks in S_a - 1 **do**
    for $j = 1$ to Number of tasks in S_a - 2 **do**
      Move the position of a pair of tasks Y_i and Y_{i+1} to the position of task Y_j
      Use *split scheme* to calculate the chromosome's fitness
    end for
  end for
  $S_a^2 \leftarrow$ The chromosome with the maximum fitness
  //Swap
  for $i = 1$ to Number of tasks in S_a - 1 **do**
    for $j = i + 1$ to Number of tasks in S_a **do**
      Swap the position of tasks Y_i and task Y_j
      Use *split scheme* to calculate the chromosome's fitness
    end for
  end for
  $S_a^3 \leftarrow$ The chromosome with the maximum fitness
  Keep the best individual among S_a^1, S_a^2 and S_a^3 as S_b

Thereafter, the algorithm is used to search for the best solution for updated instance and the vehicles continue serving again based on new schedules.

IV. Experimental Results

In this section, the performance of the proposed MASDC algorithm is studied on four dynamic problems taken from [15]. First one is a 10-node example which is used to visually illustrate the results from the algorithm. The remaining three examples are undirected, directed and mixed graphs containing 100 nodes each. For each of the 100-node examples, the instance resembles a static CARP at the beginning time instant t_0, and these are also solved using state of the art static CARP algorithm MAENS [16] for comparison.

A. 10-node example

The example contains 2 vehicles with different capacities (460 and 480) located at different depots (node 1 and node 3). The speed of the vehicles is 30km/h. I_0 is the instance at time t_0 (i.e., the beginning), and then there are interruptions/changes detected at time t_1 = 1 min and time t_2 = 3 min. So two updated instances I_1 and I_2 are generated at time t_1 and t_2. The features of the example are presented below:

- I_0: As shown in Figure 3(a), I_0 contains 10 nodes and 14 roads. There are 5 tasks to be served, which are the edges 1-4, 1-7, 4-8, 7-8 and 8-10. Vehicle 1 and vehicle 2 are both available for service.
- I_1: As shown in Figure 3(b), I_1 contains 10 nodes and 13 roads. The road 2-4 is inaccessible due to temporary

road maintenance. The demand of one existing task edge 8-10 is increased from 100 to 102. Besides the existing tasks, there are 5 new requested tasks which are edges 1-6, 1-8, 4-10(two lanes) and 5-8. Vehicle 1 is broken-down.

- I_2: As shown in Figure 3(c), I_2 contains 10 nodes and 12 roads. The roads 8-10 and 2-10 assumed to be inaccessible, while 2-4 is accessible again and requested as a task. The task edge 8-10 is canceled as road is inaccessible and task edge 1-6 is canceled as customer requirement. Besides, a new task edge 3-4 is added in. The demand of task edge 1-8 is decreased from 104 to 101. Vehicle 1 is fixed and located at its depot. Both vehicles are available to service.

Twenty independent runs are done using MASDC on the 10-node example described above. A limited budget of 2000 evaluations is used to solve the problem for each time instant $(t_0, t_1 \ldots t_{G_{max}})$. Population size (P_s) is set to 10. Local search is applied with a probability r_{LS} = 0.2. The experimental results are presented in Table I.

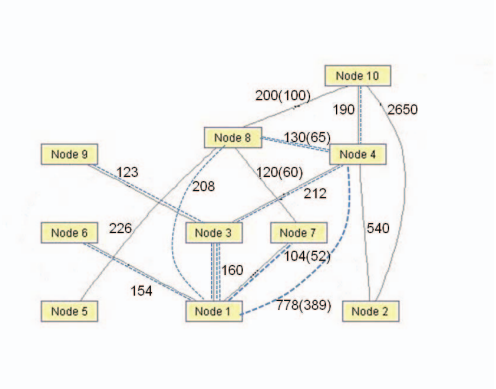
TABLE I

STATISTICS FOR TOTAL DISTANCE TRAVELED FOR 10-NODE EXAMPLE USING MASDC ACROSS 20 RUNS

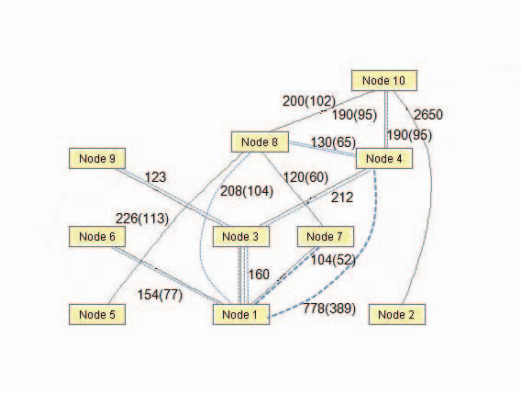
Instance	MASDC			
	Min.	Max.	Mean	Std.
10-node example	5310	6630	5824.9	418.3

The details of the best out of 20 runs are described below:

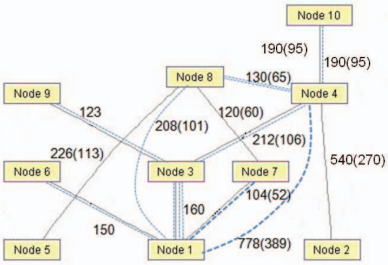
- I_0: Based on t_0, the routes of vehicle 1 and 2 are calculated as 1-8-10-4-8-7-1 and 3-1-4-3 respectively. The tasks served until t_1 are shown in Figure 4(a). Task edges 4-8, 7-8, 1-7 and 1-4 are not served yet. In terms of the condition of the vehicles, the service distance for vehicle 1 is 408 and it stops at node 10 (and has broken down); while the service distance for vehicle 2 is 160 and it stops at node 1 with a residual capacity 480.
- I_1: The updated instance at time t_1 and progression up to t_2 is shown in Figure 4(b). During this time, only one vehicle (vehicle 2) is active. It starts from node 1 and goes through tasks 1-6, 1-8, 8-4, 4-10, 10-4, then goes back to its depot node 3 to get refilled. Then it stops at node 4. At this instant, tasks 5-8, 7-8, 1-7, 1-4 and 3-4 are remaining. The service distance for vehicle 2 is 1450 and it stops at node 4 with full capacity 480.
- I_2: The updated instance at time t_2 and the route of the vehicles to serve the remaining tasks is shown in Figure 4(c). Both vehicles are available for service. Vehicle 1 starts at its depot with full capacity and vehicle 2 starts at node 4 with its full capacity of 480. The route for vehicle 1 is 1-4-8-1 (go back to depot to refill)-8-5-8-7-1 with a distance of 2000 and for vehicle 2 is 4-2-4-3 with a distance of 1292. All requested tasks are served and the vehicles return to their depots. The total distance traveled is 5310.



(a) I_0



(b) I_1

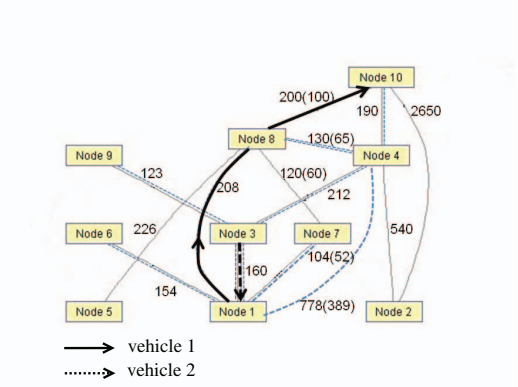


(c) I_2

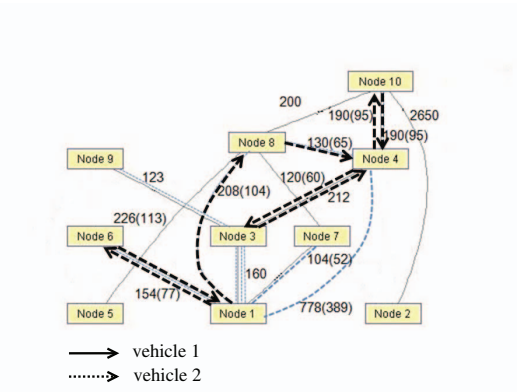
Fig. 3. The graphs for 10-node example

B. 100-node examples

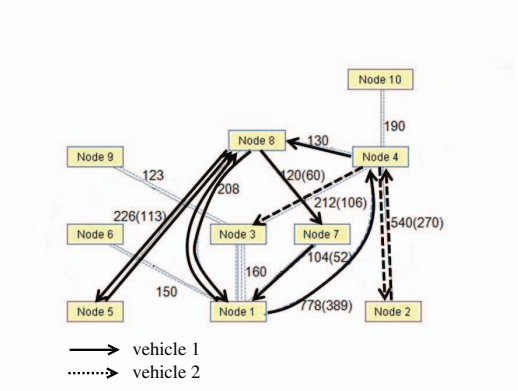
The performance of MASDC is further tested on three 100-node examples. The features of the examples are listed in Table II. Population size (P_c) is set to 30, and maximum evaluations allowed is set to 20,000 for each time step. Local search is applied with a probability r_{LS} = 0.2. Twenty independent runs are performed for each instance. At t_0, for each of these problems, the vehicles have the same capacity, are located at same depot, and the total demand is less than the total capacity. Hence the solution for this instant can also be obtained using a static CARP algorithm. A comparison



(a) I_0



(b) I_1



(c) I_2

Fig. 4. The best solution obtained using MASDC for 10-node example

is presented for the instance at t_0 between the presented algorithm and the state of art MAENS [16] algorithm. For the remaining instants (t_1 and t_2), the same conditions don't hold and they cannot be solved using static CARP methods. Hence for those two instants, only the results obtained using proposed MASDC are presented. The interruptions/changes happen at time t_1 = 4 mins and t_2 = 6 mins.

1) I_0: MASDC and MAENS Comparison: The results obtained for the three instances at t_0 are presented in Table III. In the table, the number of nodes are denoted as $|V|$, edges as $|E|$, number of one-way roads as $|OWR|$, number of vehicle as N_V, number of tasks as $|Y|$ and vehicles capacity as C. The progress plots of the median run for each instance are presented in Figure 5. As reflected from the values in Table III, MASDC on an average is able to achieve lower total distance values across multiple runs. The best values obtained using MASDC are also observed to be better than those obtained using MAENS.

TABLE II

100-NODE DYNAMIC INSTANCES: INSTANCE 1 (UNDIRECTED), INSTANCE 2 (DIRECTED) AND INSTANCE 3 (MIXED)

Output parameters	Instance 1			Instance 2			Instance 3		
	t_0	t_1	t_2	t_0	t_1	t_2	t_0	t_1	t_2
Nodes	100	100	99	100	100	100	100	100	100
Roads	164	148	139	328	295	279	208	187	177
One-way roads	0	0	0	33	27	31	6	6	4
Vehicles	8	7	4	8	7	4	8	7	4
Capacity	244	244	244	450	450	450	319	319	319
Tasks	16	15	13	33	25	28	21	19	21
Depot	2	2	2	2	2	2	2	2	2
D_1	3	6	6	3	3	3	3	6	4
D_2	2	18	26	2	4	4	2	12	20
D_3	62	52	52	62	62	63	62	58	53
D_4	30	22	13	30	28	28	30	24	22
D_5	3	2	3	3	3	2	3	0	1
ML_1	50	49.32	49.64	50	51.19	51.61	50	52.41	49.72
ML_2	43.29	43.24	43.88	42.99	41.69	40.5	42.79	41.18	41.48
ML_3	3.05	3.38	3.6	3.05	3.05	3.58	2.88	2.67	3.39
ML_4	3.66	4.05	2.88	3.96	4.07	4.3	4.33	3.74	5.08
$Dist_s$	79.88	79.73	82.01	80.18	80.34	81.72	80.29	80.21	80.23
$Dist_m$	15.24	14.86	15.11	14.94	14.58	14.34	14.9	14.97	16.38
$Dist_l$	4.88	5.41	2.88	4.88	5.08	3.94	4.81	4.81	3.39

2) Overall Results using MASDC: After the vehicles start serving the tasks on I_0, the changes occur at t_1 and t_2. The updated instances are solved using MASDC, and the total distance traveled by all vehicles are reported in Table IV. The statistics shown are calculated over 20 independent runs. Given the unavailability of existing methods to solve such problems, a comparison is not possible with other methods currently.

TABLE IV

TOTAL DISTANCE TRAVELED FOR DYNAMIC INSTANCES USING MASDC ACROSS 20 RUNS

	Min.	Max.	Mean	Std.
Instance 1	57451	65408	62308.4	1839.3
Instance 2	77011	85973	81017	2431.2
Instance 3	88538	102691	94136.3	4184.4

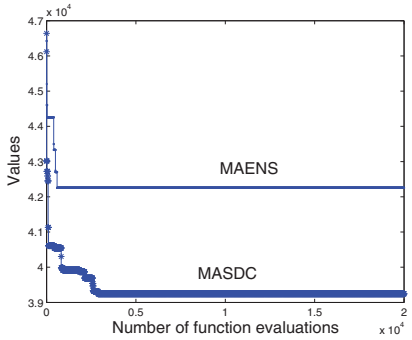
V. CONCLUSIONS

DCARP is a challenging problem which is of great interest in both research and industry due to its applicability in many real world problems. In this study, a memetic algorithm, MASDC, is proposed for solving DCARPs with variations in realistic factors such as vehicle availability, road accessibility, added/canceled tasks, traffic congestion and demands. MASDC is first of its kind to be able to solve problems with heterogeneous vehicles, different starting depots as well as cases with demands exceeding total capacity. The

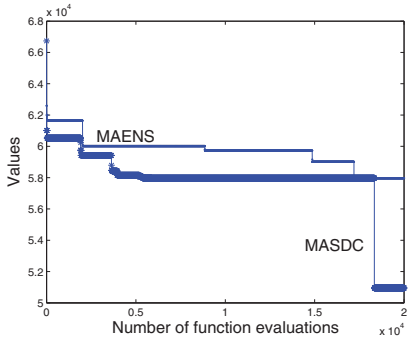
TABLE III

SOLUTIONS OBTAINED USING MAENS AND MASDC FOR 100-NODE EXAMPLES AT t_0 ($I0_1$, $I0_2$ AND $I0_3$)

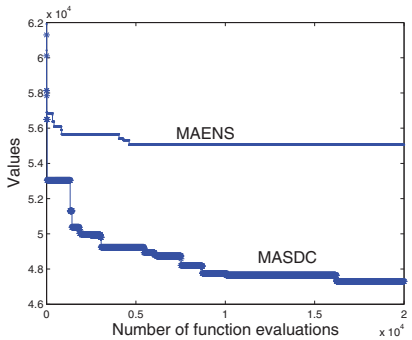
Instance	$\|V\|$	$\|E\|$	$\|OWR\|$	N_V	$\|Y\|$	C	MAENS				MASDC			
							Min.	Max.	Mean	Std.	Min.	Max.	Mean	Std.
Undirected graph $I0_1$	100	164	0	8	16	244	40709	44580	42400.1	1062.4	**39246**	39643	**39314.8**	126.4
Directed graph $I0_2$	100	328	33	8	33	450	55225	60982	58323.4	1650.9	**48836**	51652	**50639.1**	977.9
Mixed graph $I0_3$	100	208	6	8	21	319	53914	56915	55215.3	922.1	**46503**	48074	**47260.8**	492.9



(a) Median results for Instance 1 (undirected graph)



(b) Median results for Instance 2 (directed graph)



(c) Median results for Instance 3 (mixed graph)

Fig. 5. Median results for initial instance I_0 of three 100-node examples

algorithm is also capable of solving traditional static CARP instances efficiently. The working of the proposed algorithm is illustrated using a 10-node example as well as three 100-node examples of undirected, directed and mixed nature. The results for static case are compared with the state of the art algorithm MAENS to demonstrate the efficacy of the algorithm. MASDC results are competitive and the algorithm shows great potential for solving realistic DCARP problems.

REFERENCES

[1] V. Pillac, M. Gendreau, C. Guéret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, 2012.

[2] M. Tagmouti, M. Gendreau, and J.-Y. Potvin, "A dynamic capacitated arc routing problem with time-dependent service costs," *Transportation Research Part C: Emerging Technologies*, vol. 19, no. 1, pp. 20–28, 2011.

[3] H. Handa, L. Chapman, and X. Yao, "Dynamic salting route optimisation using evolutionary computation," in *IEEE Congress on Evolutionary Computation (CEC)*, vol. 1. IEEE, 2005, pp. 158–165.

[4] K. A. De Jong, "Jong: An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, PhD thesis, University of Michigan, Dissertation Abstracts International. 36 (10), 1975.

[5] G. Ulusoy, "The fleet size and mix problem for capacitated arc routing," *European Journal of Operational Research*, vol. 22, no. 3, pp. 329–337, 1985.

[6] M. Liu and T. Ray, "Efficient solution of capacitated arc routing problems with a limited computational budget," in *AI 2012: Advances in Artificial Intelligence*. Springer, 2012, pp. 791–802.

[7] B. Golden, J. DeArmon, and E. Baker, "Computational experiments with algorithms for a class of routing problems," *Computers & Operations Research*, vol. 10, no. 1, pp. 47–59, 1983.

[8] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[9] I. Oliver, D. Smith, and J. Holland, "A study of permutation crossover operators on the traveling salesman problem," in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*. L. Erlbaum Associates Inc., 1987, pp. 224–230.

[10] H. Mühlenbein and D. Schlierkamp-Voosen, "Analysis of selection, mutation and recombination in genetic algorithms," in *Evolution and biocomputation*. Springer, 1995, pp. 142–168.

[11] Michalewicz, "Repair algorithms," *Evolutionary computation 1, basic algorithms and operators*, pp. 56–61, 2000.

[12] J. Bean, "Genetic algorithms and random keys for sequencing and optimization," *ORSA journal on computing*, vol. 6, no. 2, pp. 154–160, 1994.

[13] F. Samanlioglu, W. Ferrell, and M. Kurz, "A memetic random-key genetic algorithm for a symmetric multi-objective traveling salesman problem," *Computers & Industrial Engineering*, vol. 55, no. 2, pp. 439–449, 2008.

[14] P. Lacomme, C. Prins, and W. Ramdane-Cherif, "Competitive memetic algorithms for arc routing problems," *Annals of Operations Research*, vol. 131, no. 1, pp. 159–185, 2004.

[15] M. Liu, H. K. Singh, and T. Ray, "A benchmark generator for dynamic capacitated arc routing problems," in *IEEE Congress on Evolutionary Computation (CEC)*, 2014, accepted.

[16] 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.