Efficient Solution of Capacitated Arc Routing Problems with a Limited Computational Budget

Min Liu and Tapabrata Ray

School of Engineering and Information Technology,
University of New South Wales, Northcott Drive, Canberra, Australia
min.liu@student.adfa.edu.au, T.Ray@adfa.edu.au

Abstract. Capacitated Arc Routing Problem (CARP) is a well known combinatorial problem that requires the identification of the minimum total distance travelled by a set of vehicles to service a given set of roads subject to the vehicle's capacity constraints. While a number of optimization algorithms have been proposed over the years to solve CARP problems, all of them require a large number of function evaluations prior to its convergence. Application of such algorithms are thus limited for practical applications as many of such applications require an acceptable solution within a limited time frame, e.g., dynamic versions of the problem. This paper is a pre-cursor to such applications, and the aim of this study is to develop an algorithm that can solve such problems with a limited computational budget of 50,000 function evaluations. The algorithm is embedded with a similarity based parent selection scheme inspired by the principles of multiple sequence alignment, hybrid crossovers, i.e., a combination of similarity preservation schemes, path scanning heuristics and random key crossovers. The performance of the algorithm is compared with a recent Memetic algorithm, i.e., Decomposition-Based Memetic Algorithm proposed in 2010 across three sets of commonly used benchmarks (gdb, val, egl). The results clearly indicate the superiority of performance across both small and large instances.

Keywords: Capacitated arc routing problem, memetic algorithm, heuristic, random key representation, multiple sequence alignment.

1 Background

The capacitated arc routing problem (CARP) [12] can be described as follows: Assume a road network represented using a undirected graph $G = (V, E)$ with a set V of v nodes and a set E of e edges. The set of p tasks (required edges) $T = (t_1, ..., t_p)$ needs to be served by a fleet of vehicles which are associated with a same capacity C and located at a depot s ($s \in V$). Each edge is associated with a distance $d(i) \geq 0 (i = 1, 2, ..., e)$ and a demand $q(i) \geq 0 (i = 1, 2, ..., e)$. A demand of $q(i)$ equal to zero indicates that the edges do not need service. The objective is to find the set of vehicle trips using minimum total distance D, such that each vehicle trip starts and ends at the depot s, each required edge is serviced by one single trip, and the total demand handled by any vehicle must not exceed its capacity C. The distance of a trip includes the distance of its serviced tasks and the distance traveled through edges that are not serviced.

M. Thielscher and D. Zhang (Eds.): AI 2012, LNCS 7691, pp. 791–802, 2012.

Table 1. Mathematical Symbols used in this paper

Symbol	Meaning
Q	Population
w	Population size
S	Solution
N	Number of trips
C	Capacity of vehicle
e	Number of edges
d_i $(i = 1, ..., e)$	Distance of each edge
q_i $(i = 1, ..., e)$	Demand of each edge
T	A set of tasks
s	Depot of the vehicles
p	Number of tasks
D	Total distance, i.e., fitness of the solution

The symbols used in the paper are provided in Table 1.

Arc routing problems [7, 9, 10] are classic combinatorial optimization problems, wherein the aim is to find a route with minimum total distance subject to the set of predefined constraints. CARP is a practical form of an arc routing problem, in which a fleet of homogeneous (same capacity) vehicles needs to service the set of demands associated with the edges. CARP is known to be NP-hard and thus application of exact optimization methods are still limited small problems (20-30 edges) [12]. Although exact methods can solve some large instances of CARP, the computational cost is exorbitantly large, e.g., the branch-and-cut-price algorithm required around 20000 seconds [16] to solve a problem with 190 edges. In most real-world applications, it is necessary to obtain a solution within a given time budget. Different heuristic based approaches have been proposed over the years to deal with such problems, which include augment-merge [12], path-scanning [11, 21], construct-and-strike [19], Ulusoy's tour splitting [23], argment-insert [20] etc. Metaheuristic based approaches have also been proposed such as Simulated Annealing [8], Tabu Search [17], Variable Neighborhood Search [22], Guided Local Search [4], Genetic Algorithm [18], Evolutionary Algorithm (EA) [24], Ant Colony Optimization [1] and more recently hybrids such as Memetic Algorithm (MA) [13,22] which is an intelligent combination of a genetic algorithm and a local search.

A memetic algorithm with extended neighborhood search (referred as MAENS) [22] was introduced by Yi Mei *et al* in 2010 and is one of the most efficient algorithms developed till date. However, the number of function evaluations required (approximate from 4.0×10^7 to 3.0×10^9) are still far more than what can be afforded for practical problems such as for dynamic CARPs. The performance of any optimization algorithm is largely dependent on the mechanisms of parent selection, method of recombination and local search strategies. The proposed algorithm is realized using a memetic algorithm as its baseline form. The parent selection is based on a similarity measure inspired by multiple sequence alignment while the recombination process is a hybrid consisting of path scanning heuristics and random key crossovers. The performance of the proposed method has been evaluated on three sets of CARP instances (gdb, val, egl) that contain a total of 81 instances and compared with MAENS [22]. Section 2 describes the proposed Memetic Algorithm (MA) and Section 3 describes the numerical experiments and results. The conclusions are summarized in Section 4.

2 Proposed Algorithm

The algorithm maintains a population of solutions which are evolved over generations. Fitter individuals are paired with its partner based on a multiple sequence alignment inspired similarity measure. Such a scheme identifies pairs of solutions that have the maximum number of common vehicle trips which is then inherited by the child solution. Such a process is similar to inheriting good building blocks from parents in the context of genetic algorithms. As for the remaining set of demand edges, path scanning heuristic is used. Unique solutions are always maintained in a population and in the event, the number of unique solutions is less than the population size, solutions are generated using random key crossovers. In order to further improve a child solution, a neighborhood based local search is invoked with a given probability. The pseudocode of the algorithm is presented in Algorithm 1.

2.1 Solution Representation and Generation of the Initial Population

Firstly, the undirected graph of the road network is transformed to a directed graph, i.e., each edge is represented as two directed arcs in opposite directions and each of these arcs have the same edge ID. A solution(chromosome) for CARP is represented as a list of edge IDs. The set of w chromosomes are generated to form the initial population Q using a set of edge IDs using path-scanning (PS) heuristic without considering the capacity constraints (Rule 1: maximize the distance $dist(t, s)$ from task t to depot; Rule 2: minimize the distance $dist(t, s)$; Rule 3: maximize the yield $q(t)/d(t)$, i.e., the ratio of demand/distance for each task; Rule 4: minimize the yield $q(t)/d(t)$; Rule 5: use Rule 1 if the vehicle's capacity is less than half-full, else use Rule 2) [11]. Non-unique solutions are replaced with randomly generated chromosomes. Then the chromosomes are *(*split) using Ulusoy heuristic [13] which identifies the locations of the split that results in a minimum D while satisfying capacity constraints of the vehicles. This D is assigned as the fitness of the chromosome. The individuals of the population are sorted based on their fitness.

2.2 Multiple Sequence Alignment Inspired Selection

Parent selection is an important element in any population based search strategy. There are different variants of parent selection such as through the use of roulette wheel, binary tournament, random selection etc [24]. Fundamentally, all such processes aim to generate child solutions which inherit good building blocks from fitter parents. In the context of molecular biology [14], scientists are faced with the challenge of aligning multiple DNA sequences. Each DNA sequence is a set containing the base elements, i.e., A, C, T or G, and similarity between two sequences are sought via maximization

Algorithm 1. Proposed Algorithm

REQUIRE: Population size (w), Maximum number of function evaluations (MFE), Local search probability r_{LS}

Generate an initial population Q
while Number of solutions evaluated $\leq MFE$ **do**
    Apply MSA inspired selection operator to select $w/2$ pairs of parents
    Save the common trips of each pair of parents, keep the different part of each pairs, i.e., a set of edges that contains different trips, as $P*$
    w solutions P_c are generated by using path-scanning heuristic (PS) without considering capacity constraints on each edge in $P*$ twice
    Keep the unique solutions in P_c
    if Number of $P_c \leq w$ **then**
        for i = 1:w-Number of P_c **do**
            Apply roulette wheel to select two solutions P_{c1} and P_{c2} in P_c
            Apply Random Key Crossover on P_{c1} and P_{c2} to generate a child $Child$
            Insert $Child$ into P_c
        end for
    end if
    Sort P_c
    Select the best one in P_c as C_1 (if the best one is same (all generated solutions are worse than the best in P_c), using roulette wheel instead)
    if $m \leq r_{LS}$; m is a random number [0,1] **then**
        Apply local search to C_1 to generate C_3
        if C_3 is not a clone of any chromosome in Q **then**
            Insert C_3 into Q
        else if C_1 is not a clone of any chromosome in Q **then**
            Insert C_1 into Q
        end if
    else if C_1 is not a clone of any chromosome in Q **then**
        Insert C_1 into Q
    end if
    Sort the chromosomes in Q and keep the best w solutions in Q
end while

of the matching score. In the current context of a population of solutions sorted based on fitness, a partner of solution P_1 is the solution that has maximum number of vehicle trips in common. In the event there are multiple potential partners (i.e., with the same number of trips in common), a random partner is chosen among them.

2.3 Hybrid Recombination

A hybrid recombination scheme is used in this study. The recombination process identifies trips that are common in both the parents and maintains the same for the child solution. As for the remaining set of edges, a path-scanning heuristic without considering the capacity (PS) is used to generate the giant trip. By applying PS twice on the same set of remaining edges, two offsprings are generated. After all w child solutions are created, unique solutions are maintained and in the event the number of unique solutions is less than the size of the population, a random key crossover [2] is used to generate new solutions. From the set of w parent solutions and newly generated w child solutions, the best w individuals are maintained in Q.

The process of random key crossover is illustrated below for completeness. Two individuals P_{c1} and P_{c2} are identified using a roulette wheel based selection. Seven random numbers (one for each task in P_{c1}) are drawn uniformly from (1, 3000) and for the discussion assume them as follows : (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). It is important to highlight that the random numbers used in P_{c1} and P_{c2} should be unique. Following the standard form of two point crossover 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 offsprings generated by crossover via *random keys* are guaranteed to be feasible and a random one is selected. The pseudocode of the process is presented below.

Algorithm 2. Modifed Random Key Crossover

for $i = 1 \rightarrow w - Number of P_c$ **do**
  Apply roulette wheel to select two individual P_{c1} and P_{c2}
  for $j = 1 \rightarrow 2$ **do**
    Randomly select h numbers $\subseteq (1, 3000)$ {Number of total tasks in P_{cj}}
    Encode $P_{cj} \Longrightarrow R_i$
    Randomly select two integer numbers $m_{point1} \subseteq [1, h-1]$ and $m_{point2} \subseteq [1, h-1]$ to split R_i
    while $m_{point1} = m_{point2}$ **do**
      Randomly select a integer number $m_{point2} \subseteq [1, h-1]$
    end while
    $R_i \Longrightarrow (part_i^1*, part_i^2*, part_i^3*)$
  end for
  Swap $(part_1^2* \Longleftrightarrow part_2^2*) \Longrightarrow R_1*, R_2*$
  for $j = 1 \rightarrow 2$ **do**
    Decode $R_j* \Longrightarrow Chr_j*$
  end for
  Randomly select Chr_1* and $Chr_2* \Longrightarrow C_1$
end for

2.4 Local Search

Apart from recombination, local search plays an important role in any hybrid or memetic forms. Firstly, the child C_1 is improved using a local search with a probability r_{LS}. Three move operators have been used to perturb the solution, i.e., *single insertion*, *double insertion*, and *swap*. The best chromosome C_2 with the shortest distance is the outcome of this phase. It is very important to highlight that if the top Q is still the same after a generation, a roulette wheel selection is used to identify C_1.

The best result S from *splitting* C_1 identified from phase 1 is now improved further using a larger search domain, wherein tasks of two vehicle trips are redistributed among them using five rules of path scanning respectively resulting in five candidate part-solutions. The best candidate part-solution is inserted into the rest of the S to result in the new solution S_C. Since there are C_N^2 combinations possible, the following condition is enforced, i.e., if I:$N!/2(N-2)! < 50$, l_{times} = I, else l_{times} = 50, where l_{times} denotes the number of attempts.

The solution identified through this second phase of local search is accepted if its distance is lower than the one obtained during the first phase of local search, else the local search phase is aborted and the best solution corresponds to the one obtained in phase one. In the event, the solution obtained from the second phase of local search is better than the one obtained from the first phase, the algorithm offers one more chance to the solution to improve while moving through phase one and phase two of the local search procedure. The best solution of this phase is denoted as C_3.

The pseudocode of the local search process is presented in Algorithm 3.

Algorithm 3. Local Search Algorithm

REQUIRE: Probability $\leq r_{LS}$

Perform $single\ insertion$ operator to $C_1 \Longrightarrow C_1^1$
Perform $double\ insertion$ operator to $C_1 \Longrightarrow C_1^2$
Perform $swap$ operator to $C_1 \Longrightarrow C_1^3$
Keep the best one of C_1^1, C_1^2 and $C_1^3 \Longrightarrow C_2$
Apply $split$ method to $C_2 \Longrightarrow S = (S_1, S_2, ..., S_N)$
l_{times} = min $[I, 50]$
for $i = 1$ to l_{times} **do**
    Apply path-scanning to each pair of solutions to generate five different *part solutions*
    Select the best one of them $\Longrightarrow S_C$
    Combine S_C with the rest solutions $\Longrightarrow C_3$
    Apply $split$ method to evaluate C_3
    Update the solution if C_3 is better than C_2, $C_3 \Longrightarrow C_{new}$
end for
if Found C_{new} = true **then**
    Apply local search again on C_{new}
end if

3 Performance on Benchmarks

In this section, the computational experiments and results are discussed. The first benchmark set contains 23 instances (gdb) originally generated by DeArmon [6]. The second benchmark set is from Benavent $et\ al$ [3] which contains 34 instances (val benchment) defined using 10 different graphs. In both these instance sets, all the edges of the graphs require service. The final benchmark set [8] consists of 24 instances with large number of edges constructed from winter gritting data of Lancashire. The road networks of all the instances are undirected. Although the proposed algorithm can deal with mixed networks, we still focus on these problems since there are no mixed graph data instances in public domain.

The performance of the proposed algorithm is presented in Tables 2 3 4 and compared with the state-of-the-art algorithm MAENS [22] using a limited budget of 50,000 function evaluations(FE). For each instance, the results reported are averaged over 20 independent runs. Population size (w) is set to 30. Local search is applied with a probability r_{LS} = 0.2. In each table, the number of vertices are denoted as $|V|$, edges as $|E|$, the number of tasks as $|T|$, the average computational time as $T_c(second)$ and low bound as LB [4, 5, 15]. The average distance, standard deviation of the distance, best distance and the number of vehicles corresponding to the solution with the best distance are indicated for all the problem instances. The value in column R_s indicates the number of runs in which the worst solution obtained by the proposed algorithm in 20 runs is better than the best obtained using MAENS. A value of 20 indicates that the results obtained in all the runs of the proposed algorithm is better than the best obtained using MAENS. The progress plot of the median run for one of each problem classes are presented in Figure 1. One can clearly observe the benefits of the initialization scheme and the efficiency of the proposed approach. Detailed comparison of results of MAENS with other approaches have appeared in [22] and hence have been omitted in this paper.

Table 2 presents the results of experiments on the gdb benchmark, which consist of small networks with no more than 55 tasks. It is clear that the proposed algorithm performs significantly better than MAENS in terms of the best, average and the standard deviation when the budget is limited to 50,000FE. Table 3 and Table 4 present the results for val and egl benchmarks which represent large size network instances. It can be observed that the proposed algorithm still performs significantly better than MAENS in all aspects, i.e., best, average and the standard deviation. While this study considered the performance of the proposed algorithm with limited number of evaluations (50,000), a quick comparison with the known lower bounds reported in [13] indicates that in 63 out of 81 instances, the best solution obtained by the proposed algorithm is within 20% of the lower bound. In order to observe the performance of the algorithm for higher number of function evaluations, the algorithm was allowed to use 250,000 function evaluations and in 71 out of 81 instances the best solution was siwthin 20% of the lower bound. One can observe a signifcant improvement in performance when the allowed number of function evaluatsiona are higher. In order to study the effect of MSA based parent selection, the same algorithm was run with random parent selection. The results of MSA based parent selection are signifcanltly better than the random parent selection scheme.

Table 2. Results on the set of gdb benchmark

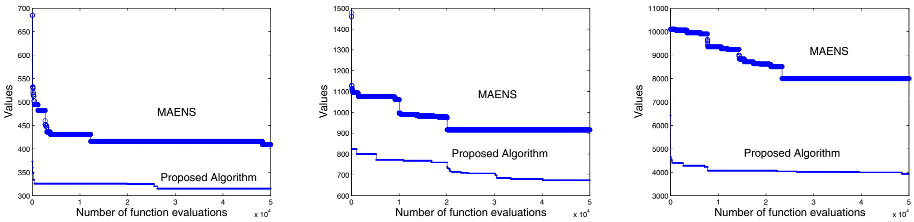
Inst.	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert T\rvert$	LB	MAENS(50,000FE)				R_s	With-MSA(50,000FE)					With-MSA(250,000FE)				Without-MSA(50,000FE)			
					$Average_1$	Std_1	D_1	N_1		$Average_2$	Std_2	D_2	N_2	$T_c(s)$	$Average_3$	Std_3	D_3	N_3	$Average_4$	Std_4	D_4	N_4
gdb1	12	22	22	316	397.9	18	372	5	30	323.5	5.6	**316**	5	20.2	319.3	4.3	316	5	329.2	6.3	316	5
gdb2	12	26	26	339	433.8	12.7	420	6	30	362.4	9.3	**345**	6	19.2	351.3	4.5	345	6	370	8.6	353	6
gdb3	12	22	22	275	394.3	7.9	382	6	30	288.1	5.6	**275**	5	18.5	281.4	4.9	275	5	294.8	7.7	281	5
gdb4	11	19	19	287	350	21.8	333	4	30	298.3	11.5	**287**	4	19.6	289.1	5.3	287	4	300	8.6	287	4
gdb5	13	26	26	377	495.9	10.7	481	6	30	400.2	9.7	**383**	6	19.8	387.2	6.6	377	6	405.6	11.3	383	6
gdb6	12	22	22	298	440.1	21.9	413	6	30	317.3	8.1	**298**	5	17.9	308.2	7	298	5	321.8	6.5	306	5
gdb7	12	22	22	325	413.5	9.3	405	6	30	333.2	7	**325**	5	19.5	328.1	3.9	325	5	341.1	9.6	325	5
gdb8	27	46	46	348	647.6	13.5	633	12	30	387.6	11.9	**364**	10	24.5	371	7.9	362	11	390.7	11.1	369	11
gdb9	27	51	51	303	568	16	541	13	30	363.8	14.8	**336**	11	22.3	343.3	7.8	331	11	374.4	18.7	339	11
gdb10	12	25	25	275	418.5	18.4	398	6	30	302.2	10.1	**288**	4	18.4	281.5	5.3	275	4	310.3	8.7	295	5
gdb11	22	45	45	395	781	15.8	756	6	30	495.4	23.4	**439**	6	20.1	446.2	17.9	419	5	503.6	21.1	444	6
gdb12	13	23	23	458	646.6	25.3	616	8	30	484	12.4	**462**	7	20.3	482.9	12.9	462	7	477.8	9.8	468	7
gdb13	10	28	28	536	626.4	7.3	617	8	30	553.5	4.8	**546**	7	20.3	548.6	1.7	544	7	557.6	5.8	548	7
gdb14	7	21	21	100	125	3.5	121	6	30	101.2	1.2	**100**	5	19.5	100.3	0.7	100	5	102.4	1.2	100	5
gdb15	7	21	21	58	67.3	2.3	66	4	30	58.5	0.9	**58**	4	20.2	58	0	58	4	58.9	1	58	4
gdb16	8	28	28	127	155.5	4.8	147	5	30	131	1.6	**129**	6	17.9	129.1	1	127	5	132.5	2	129	6
gdb17	8	28	28	91	107.6	3.1	105	5	30	91.7	0.9	**91**	5	18.1	91	0	91	5	93.3	3.1	91	5
gdb18	9	36	36	164	220.1	5.8	211	5	30	180.9	3.9	**173**	5	18.2	169.3	2.2	166	5	185	4.3	176	5
gdb19	8	11	11	55	55.4	0.5	55	3	30	55	0	**55**	3	22.8	55	0	55	3	55	0	55	3
gdb20	11	22	22	121	144.9	4.9	140	5	30	123.8	0.9	**123**	5	17.7	123	0	123	5	125.2	1.4	124	5
gdb21	11	33	33	156	218	7.1	209	8	30	165.9	3.4	**159**	7	18.4	159.4	1.5	156	6	168.9	4.5	160	7
gdb22	11	44	44	200	240.9	2.7	236	9	30	210.8	3.2	**204**	9	22	205.6	1.2	204	9	213.3	3.8	206	10
gdb23	11	55	55	233	293.5	4.2	290	13	30	253.3	4.3	**244**	12	30.3	245	2.8	237	11	258.6	6	248	13

Table 3. Results on the set of val benchmark

Inst.	$\|V\|$	$\|E\|$	$\|T\|$	LB	MAENS(50,000FE)				R_s	With-MSA(50,000FE)					With-MSA(250,000FE)				Without-MSA(50,000FE)			
					$Average_1$	Std_1	D_1	N_1		$Average_2$	Std_2	D_2	N_2	$T_c(s)$	$Average_3$	Std_3	D_3	N_3	$Average_4$	Std_4	D_4	N_4
val1a	24	39	39	173	327.5	11.8	316	2	30	215.8	12.4	**189**	3	24.4	197.9	10.9	180	3	213.9	10.11	**186**	3
val1b	24	39	39	173	324	10.8	312	4	30	216.2	9.1	**191**	4	21.7	192.2	6.2	183	4	214.3	9.6	192	4
val1c	24	39	39	245	369.3	6.7	359	11	30	267.4	6.9	**254**	10	20.3	256.6	4.1	250	9	275.6	9.2	258	10
val2a	24	34	34	227	369.8	11.9	355	2	30	300.1	26.3	**246**	2	22.6	247.9	7.4	231	2	299.8	15.7	269	2
val2b	24	34	34	259	410.8	13.1	398	4	30	315.8	16.6	**278**	3	19	282.4	7.7	264	3	319.5	17.4	278	3
val2c	24	34	34	457	589	13.3	572	10	30	499.1	11.9	**479**	8	20.3	488.2	10.7	473	8	501.6	12.9	**475**	8
val3a	24	35	35	81	137.8	4.2	133	2	30	104.6	7	**86**	2	22.8	89.6	3.4	84	2	104.7	6.5	91	2
val3b	24	35	35	87	162.5	7.2	154	3	30	112.7	7.1	**98**	4	20.2	98.2	4.2	93	3	114.9	4.3	108	3
val3c	24	35	35	138	218.7	6	210	8	30	151.3	3.6	**143**	7	17.4	148.1	2.9	144	7	155.2	5.2	146	8
val4a	41	69	69	400	796.6	7.2	788	4	30	551.9	21.4	**506**	4	31.6	539.9	20	506	4	549.3	21	507	4
val4b	41	69	69	412	844.1	15.8	827	5	30	570.3	19.3	**531**	5	38	557.6	23.8	495	4	569.4	18.6	**529**	5
val4c	41	69	69	428	778	8.7	768	7	30	596.7	21.4	**528**	6	31.2	573	18.9	528	6	598.8	16.9	557	7
val4d	41	69	69	530	922.6	16.3	907	11	30	671.2	25.4	**620**	10	33.2	638.6	19	601	9	678.7	20.7	630	9
val5a	34	65	65	423	736.1	16.2	720	5	30	590.5	18.1	**546**	4	27.6	576.4	21.6	539	3	588.5	23.4	**538**	3
val5b	34	65	65	446	808	14.2	793	5	30	613.1	22.8	**545**	5	36.4	601.6	20.8	549	4	611	17	573	5
val5c	34	65	65	474	831.3	9	822	6	30	643.5	17.7	**587**	6	27	621.3	15.9	571	6	647.6	19.5	612	6
val5d	34	65	65	577	943.7	10.5	930	12	30	729.4	22.3	**689**	9	28.1	703.3	12.3	676	10	730.2	18.6	689	10
val6a	31	50	50	223	430	11.8	412	4	30	289.8	8.3	**276**	5	26.9	275.5	13.4	253	4	292.9	9.8	**274**	4
val6b	31	50	50	233	422.7	11.3	409	5	30	300.7	12.9	**271**	5	22.1	274.1	12.9	253	5	303.1	9.6	281	5
val6c	31	50	50	317	542.2	7.9	531	11	30	372.7	10	**339**	10	24.7	346.3	8.9	330	10	373.5	12.9	354	10
val7a	40	66	66	279	638	14.5	620	4	30	363.1	12.7	**333**	4	35.3	354.6	10.4	335	4	365.1	11.4	343	4
val7b	40	66	66	283	553	12.2	540	7	30	356.2	16.1	**319**	5	27.5	339	10.6	316	5	363.1	11.5	339	5
val7c	40	66	66	334	602	14.6	586	11	30	403.6	14.4	**371**	10	27.6	381.7	11.8	362	9	421.9	18.1	385	11
val8a	30	63	63	386	783.1	16.8	762	5	30	548.1	18	**516**	3	29.2	527	16.9	501	4	554.8	15.6	524	5
val8b	30	63	63	395	721.7	13.5	706	5	30	568.9	24.1	**494**	4	28.5	536.3	26.2	474	4	581.9	16.2	547	5
val8c	30	63	63	521	895.3	14.9	878	12	30	678.7	23.1	**623**	11	33.7	636.5	14.1	610	10	686.3	28.5	635	10
val9a	50	92	92	323	748.8	16.7	726	4	30	431	15.7	**399**	4	54	405.6	10.1	392	4	435.1	12.5	402	4
val9b	50	92	92	326	785	16.6	761	5	30	443.8	12.1	**416**	6	58.1	431.1	9.6	402	6	441.7	10.4	416	6
val9c	50	92	92	332	666.1	13	652	6	30	453	11.9	**420**	6	58.2	423.3	10.1	399	6	456	8.4	436	6
val9d	50	92	92	391	815	15	795	12	30	519	16.3	**480**	12	50.2	483.5	12.1	457	12	520	15	481	11
val10a	50	97	97	428	855.6	16.4	841	5	30	584	12.1	**555**	4	61.6	577.9	12.9	555	4	580.7	11.5	556	3
val10b	50	97	97	436	712.6	12.9	697	6	30	598.6	12.5	**571**	5	73.4	595.8	17.8	554	6	600.8	10.8	575	5
val10c	50	97	97	446	993.7	11.3	987	7	30	611.3	15.5	**555**	6	61.4	590.5	18.2	534	6	612.9	11	585	6
val10d	50	97	97	531	926.3	13.3	911	12	30	696.6	14.8	**656**	10	58	681.9	14	653	10	692.5	15.8	**654**	11

Table 4. Results on the set of egl benchmark

Inst.	$\|V\|$	$\|E\|$	$\|T\|$	LB	MAENS(50,000FE)				R_s	With-MSA(50,000FE)					With-MSA(250,000FE)				Without-MSA(50,000FE)			
					$Average_1$	Std_1	D_1	N_1		$Average_2$	Std_2	D_2	N_2	$T_c(s)$	$Average_3$	Std_3	D_3	N_3	$Average_4$	Std_4	D_4	N_4
egl-e1-A	77	98	51	3548	7856	187	7618	6	30	3926	95.6	**3729**	5	20.7	3783	75.7	3672	5	4029	147	3766	5
egl-e1-B	77	98	51	4498	8274	194.1	8026	9	30	4852	78.3	**4681**	7	24.5	4728	78.9	4603	7	4945	158.4	4683	7
egl-e1-C	77	98	51	5566	8723	157.2	8563	10	30	5992	64.1	**5850**	10	24.5	5876	65	5742	10	6034	89.9	5896	10
egl-e2-A	77	98	72	5018	8335	100.2	8211	8	30	5800	150.9	**5486**	7	27	5499	128.3	5226	7	5763	161.5	5419	7
egl-e2-B	77	98	72	6305	8799	127.8	8630	10	30	6936	139.7	**6728**	10	26.6	6733	90.8	6563	10	7035	164.6	6743	10
egl-e2-C	77	98	72	8243	14433	102.9	13650	16	30	8959	100.9	**8716**	15	26.5	8778	86.4	8600	14	9024	190.8	8750	14
egl-e3-A	77	98	87	5898	12545	102	10960	9	30	6752	100.5	**6422**	8	43.6	6530	120.3	6297	8	7005	262.7	6462	8
egl-e3-B	77	98	87	7704	13720	142.1	12333	14	30	8659	127.4	**8267**	13	44.9	8767	263.9	8296	12	8383	111.9	8172	12
egl-e3-C	77	98	87	10163	16166	199.5	15930	18	30	11163	149.4	**10827**	17	44.2	10968	99.5	10793	17	11217	198.9	10827	17
egl-e4-A	77	98	98	6408	10381	162.2	10200	12	30	7532	234.9	**7024**	9	53.3	7319	184.9	7000	9	7589	228.2	7131	9
egl-e4-B	77	98	98	8884	15440	222	15201	15	30	10106	169.4	**9613**	15	55.3	9897	170.3	9562	14	10183	348.7	**9608**	14
egl-e4-C	77	98	98	11427	18577	206.6	18265	22	30	12735	201.5	**12210**	20	55.1	12521	178.9	12080	20	12811	222.9	12542	21
egl-s1-A	140	190	75	5018	8210	176.4	7964	8	30	5734	110.5	**5433**	7	41.6	5564	88.3	5368	7	5827	170.8	5449	7
egl-s1-B	140	190	75	6384	9823	177.3	9633	11	30	6940	124.7	**6721**	10	33.8	6854	81.9	6631	10	7081	163.1	**6718**	10
egl-s1-C	140	190	75	8493	14505	198.2	14212	14	30	8971	122.1	**8715**	14	33.9	8786	95.8	8668	14	9252	248.9	8792	15
egl-s2-A	140	190	147	9824	19136	176	18965	16	30	11752	152.2	**11372**	15	95.4	11545	181.2	10987	14	11920	153.1	11497	14
egl-s2-B	140	190	147	12968	20082	228.1	19869	22	30	15004	159.6	**14706**	21	102.7	14784	160.3	14387	21	15082	268.7	14744	21
egl-s2-C	140	190	147	16353	21523	142.5	21321	29	30	18645	194.4	**18291**	29	100.1	18404	179.5	18115	29	18765	258.8	**18249**	29
egl-s3-A	140	190	159	10143	19290	226.7	18927	16	30	12124	158	**11792**	15	116	12011	139.3	11654	15	12222	218.8	11802	15
egl-s3-B	140	190	159	13616	22565	145.3	20563	23	30	15900	189.3	**15505**	23	111.7	15739	179.9	15433	22	15893	223.7	15564	24
egl-s3-C	140	190	159	17100	25643	244	25362	31	30	19464	207.4	**19069**	30	108	19291	122.2	18909	31	19464	199.6	**18959**	30
egl-s4-A	140	190	190	12143	18733	210.1	18520	20	30	14956	247.8	**14334**	20	176.3	14771	317.5	13991	19	14966	264.3	14452	19
egl-s4-B	140	190	190	16093	24411	206.7	24122	30	30	19041	194.5	**18533**	28	173.9	18900	178.2	18449	29	19075	157.2	18621	28
egl-s4-C	140	190	190	20375	26730	167.9	26541	37	30	23669	263.1	**23240**	39	168	23489	275.4	22926	38	23618	174.4	23251	38



(a) gdb6 (b) val8c (c) egl-e1-A

Fig. 1. The rate of convergence

4 Conclusion

In this paper, a memetic algorithm is introduced that relies on the use of multiple sequence alignment inspired parent selection scheme, a hybrid recombination strategy and a local search. Results on both small and large scale instances with limited number of solution evaluations indicate the applicability of the proposed algorithm for dynamic CARP problems, where one is interested in identifying an acceptable quality of solution within a short time. Although in reality, computational time is of primary interest, we focus here on a more objective performance indicator, i.e., number of function evaluations which is independent of the computing platform and skills of implementation. The proposed algorithm obtained high quality solutions to 81 well studied CARP instances within 50,000 function evaluations and all of which are better than the existing state-of-the-art approach based on MAENS. The performance of the algorithm is also presented for 250,000 function evaluations for further studies. Apart from the convergence observed in the objective function space, the detailed performance on its components are analyzed and discussed. In future, we would incorporate an adaptive strategy which allocates/redistributes function evaluations to local search and recombination strategies based on their success which is likely to further improve the effectiveness of the algorithm.

References

1. Bautista, J., Fernández, E., Pereira, J.: Solving an urban waste collection problem using ants heuristics. Computers & Operations Research 35(9), 3020–3033 (2008)
2. Bean, J.: Genetic algorithms and random keys for sequencing and optimization. ORSA Journal on Computing 6(2), 154–160 (1994)
3. Benavent, E., Campos, V., Corberán, A., Mota, E.: The capacitated arc routing problem: lower bounds. Networks-New York 22, 669–669 (1992)
4. Beullens, P., Muyldermans, L., Cattrysse, D., Van Oudheusden, D.: A guided local search heuristic for the capacitated arc routing problem. European Journal of Operational Research 147(3), 629–643 (2003)
5. Brandão, J., Eglese, R.: A deterministic tabu search algorithm for the capacitated arc routing problem. Computers & Operations Research 35(4), 1112–1126 (2008)

6. DeArmon, J.: A comparison of heuristics for the capacitated chinese postman problem. Master's thesis (University of Maryland, College Park, MD) (1981)
7. Dror, M.: Arc routing: theory, solutions, and applications. Springer, Netherlands (2000)
8. Eglese, R.: Routeing winter gritting vehicles. Discrete Applied Mathematics 48(3), 231–244 (1994)
9. Eiselt, H., Gendreau, M., Laporte, G.: Arc routing problems, part i: The chinese postman problem. Operations Research, 231–242 (1995)
10. Eiselt, H., Gendreau, M., Laporte, G.: Arc routing problems, part II: The rural postman problem. Operations Research, 399–414 (1995)
11. Golden, B., DeArmon, J., Baker, E.: Computational experiments with algorithms for a class of routing problems. Computers & Operations Research 10(1), 47–59 (1983)
12. Golden, B., Wong, R.: Capacitated arc routing problems. Networks 11(3), 305–315 (1981)
13. Lacomme, P., Prins, C., Ramdane-Cherif, W.: Competitive memetic algorithms for arc routing problems. Annals of Operations Research 131(1), 159–185 (2004)
14. Lecompte, O., Thompson, J., Plewniak, F., Thierry, J., Poch, O.: Multiple alignment of complete sequences (macs) in the post-genomic era. Gene 270(1-2), 17–30 (2001)
15. Longo, H., de Aragâo, M., Uchoa, E.: Solving capacitated arc routing problems using a transformation to the cvrp. Computers & Operations Research 33(6), 1823–1837 (2006)
16. Martinelli, R., Pecin, D., Poggi, M., Longo, H.: A branch-cut-and-price algorithm for the capacitated arc routing problem. Experimental Algorithms, 315–326 (2011)
17. 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)
18. Morgan, M., Mumford, C.: A weight-coded genetic algorithm for the capacitated arc routing problem. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 325–332. ACM (2009)
19. Pearn, W.: Approximate solutions for the capacitated arc routing problem. Computers & Operations Research 16(6), 589–600 (1989)
20. Pearn, W.: Augment-insert algorithms for the capacitated arc routing problem. Computers & Operations Research 18(2), 189–198 (1991)
21. Santos, L., Coutinho-Rodrigues, J., Current, J.: An improved heuristic for the capacitated arc routing problem. Computers & Operations Research 36(9), 2632–2637 (2009)
22. 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)
23. Ulusoy, G.: The fleet size and mix problem for capacitated arc routing. European Journal of Operational Research 22(3), 329–337 (1985)
24. Xing, L., Rohlfshagen, P., Chen, Y., Yao, X.: An evolutionary approach to the multidepot capacitated arc routing problem. IEEE Transactions on Evolutionary Computation 14(3), 356–374 (2010)