CS311 Project2 Report: An Evolutionary-based Algorithm for solving Capacitated Arc Routing Problem

Zhengdong Huang *12212230*

***Abstract*—This article is a project report of CS311 Project 1. In this report, I proposed a Evolutionary-based CARP-solving Algorithm(TGA) that combines the concepts of Genetic Algorithm, Tabu Search Algorithm and Hill Climbing Algorithm. Also, I analyzed the model's complexity and conducted experiments. The results demonstrate the effectiveness of TGA with high-quality output and fast convergence speed through the combination of three algorithms.**

***Index Terms*—Evolutionary Algorithm,Capacitated Arc Routing Problem, Combinatorial optimization problem**

I. Introduction

RESEARCH on Arc Routing Problem dates back to the seminal work of the Swiss mathematician Leonhard Euler in 18th century and further formally introduced by Golden and Wong in 1981 as a generalization of the Chinese Postman Problem (CPP) for situations where vehicles have limited capacity [4]. This combinatorial optimisation problem aims to determine the optimal routing plan that satisfied the limitation. The basic form of Arc Routing Problem is the Capacity Arc Routing Problem (CARP). Given a Graph with distance and demand for each edge and a single depot, the objective of CARP is to find a routing plan of vehicles that satisfied the following condition [6]:

1. Each route starts and ends at depot.
2. Each required edge is serviced once
3. The total demand in each route do not exceed the capacity of vehicle.

Since CARP aims to minimise path costs, it can be widely used to solve real-life path planning problems. One of the most common application of CARP is to design the routes of garbage collecting truck, which is a significant part in waste collection [17]. In addition, CARP problems are widely applied in the planning of street sweeping routes, postal delivery routes, and other similar problems, where existing studies have effectively improve the quality of routes plan [15].

Beside the formal representation of Capacity Arc Routing Problem, including the mixed CARP (MCARP) that has partly-directed Graph, multi-depot CARP (MDCARP) that has multiple depots in Graph, muiti-objective CARP (MOCARP) that has multiple objectives to reach [2], [3]. Also, there is a similar problem named Capacitated Vehicle Routing Problem (CVRP) where the demand exist on vertices instead of edge. The similarities between CVRP and CARP have led many researchers to take inspiration from the solution methods of the CVRP to solve the CARP [9]

As proven by Golden and Wong [14], CARP and its variants are NP-hard problems, meaning that the optimal can't be solved in polynomial time. Therefore, the researches mainly focus on exploring deterministic or non-deterministic algorithms to approximate the optimal solution. Due to the complexity of CARP, deterministic algorithms tend to be complex and underperform, where works have proved the lower bounds of the deterministic algorithm based on dynamic programming, cutting-plane method and so on [1], [20]. In comparison, non-deterministic algorithms including evolutionary algorithms are simpler and perform better by heuristic search with stochasticity.

This paper will primarily demonstrate the effectiveness of utilizing Evolutionary-based algorithm to solve Capacity Arc Routing Problem. It will delve into the implementation of a hybrid evolutionary algorithm proposed in paper named as TGA that combines the Genetic Algorithm (GA), Tabu Searching Algorithm(TS) and Hill Climbing Algorithm(HCA) for solving the CARP, and endeavor to compare its performance with other solving algorithms. We seek to elucidate the effectiveness of our methodology and its implications for problem-solving domains.

II. Preliminary

The CARP can be formulate as follows: Consider a undirected connected simple graph $G = (V, E)$ with a vertex set V and an edge set E, with set of required edges $T \subset E$. For $\forall e \in E$, e has two attributes: cost c and demand d that satisfied the condition $c > 0$, $d \geq 0$, and the demand of e $e_d > 0$ occurs if and only if $e \in T$. The graph also contains a single depot $s \in V$. Also, we define the route r represent the path for each vehicles to serve the demand edge, with the identical capacity limitation Q for each vehicles. The cost of r,represented as $r.c$, is defined as the sum of the cost of the edges in the path, and the load of r, represented as $r.d$, is defined as the sum of the demand of the served edges in path. For a route r that pass m edges and served n edges, it is represented as a tuple of edge tuples et in form of $(v, w), v, w \in V$, and served edges $es \in E$ as follows:

$$r = (et_1, et_2, ...et_m) \ and \ \{es_1, ...e, es_n\}$$

The r satisfied the condition that $\forall et = (v, w) \in (et_1, et_2, ...et_m)$, there is a edge $e \in E$ that connects the vertices v and w. And $\forall 1 \leq i \leq m-1, e_i.w = e_{i+1}.v$. Also $\forall es \in \{es_1, ...es_n\}$, there is a $et \in (et_1, et_2, ...et_m)$ that its

representative edge e is same as es
The CARP aims to find a set of routes R that satisfied the following condition:
1. $\forall r \in R$, it start and end at the depot s i.e. $r.et_1.v = r.et_m.w = s$
2. Each required edges is served exactly once.
3. $\forall r \in R$, $r.d \leq Q$
The objective of CARP is to minimize total the cost of the routes, that is for a solution R with Rm routes, $R = r_1, ...r_m$,

$$\textbf{minimize } TC(R) = \sum_{k=1}^{Rm} r_k.c$$

III. Methodology

The algorithm's design is centered around the process of Population Initialization, Individual Selection & Mutation, and Local Search. This approach is further complemented by various optimization strategies to boost the agent's performance. The workflow of this method will be presented in subsection A, followed by a detailed explanation of the fundamental steps of the algorithm in subsection B. The performance analysis will be further detailed in subsection C.

A. General Workflow

In this article, we proposed a hybrid evolutionary CARP solving algorithm named as TGA that combines the Genetic Algorithm, Tabu Searching Algorithm and Simulated Annealing Algorithm. The algorithm mainly combines the thought of [12], [19], [21] The general workflow of the agent algorithm can be described as Fig.1.

B. Fundamental Steps in TGA

1) Preprocessing on Graph: The preprocessing steps on graphs mainly utilize Floyd's Algorithm to evaluate the minimum distance between each vertex. While solving the CARP problem, we observe that if we have determined the serving sequence of edges for a given route, we can easily calculate the cost of the route by knowing the minimum distance between each vertex. Thus, we can convert the solving objective into determine the optimal serving sequences rather than the real routes, which significantly reduce the complexity of solving process.

By above analysis, we observe that we need to calculate the minimum distance between each vertex i.e. solving Multi-source Shortest Path Problem (MSSP), which has been extensively researched. The classical solution of this problem is the Floyd-Warshall algorithm [10], the time complexity of this algorithm is $O(|V|^3)$.

Another typical approach is to run the Single-source Shortest Path Problem like Dijkstra Algorithm [5] to solve the problem, where multiple optimization on this Algorithm might speed up the preprocessing steps [18], [22]. Most of the classical optimization, like Heap-Optimized strategy [11], takes $O(|V|(|V|log|V| + |E|))$ to solve the MSSP. For a sparse graph that satisfied $|E| = O(|V|)$, the time complexity of the algorithm takes approximately $O(|V|^2 log|V|)$, which is smaller than Floyd-Warshall.However, in dense graph where

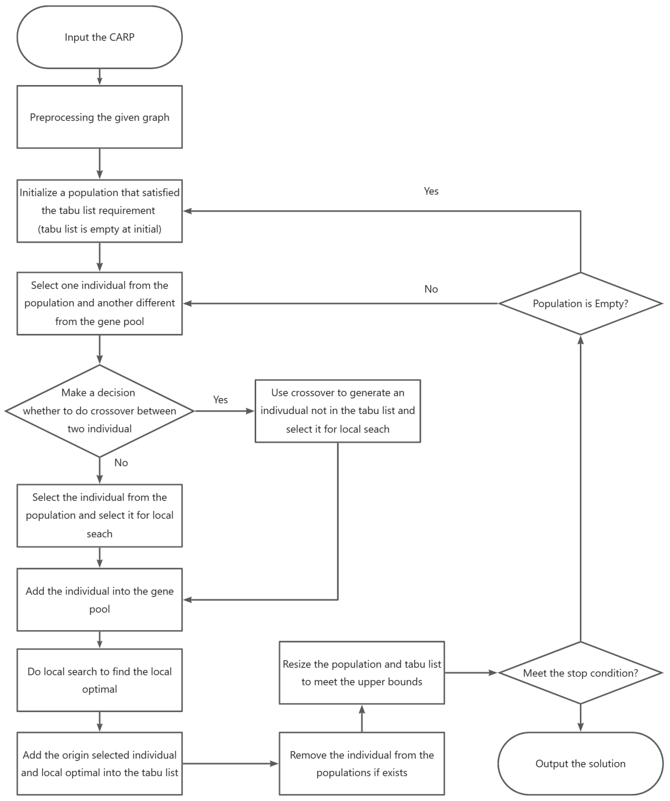

Input the CARP
Preprocessing the given graph
Initialize a population that satisfied the tabu list requirement (tabu list is empty at initial)
Yes
Select one individual from the population and another different from the gene pool
No
Population is Empty?
Make a decision whether to do crossover between two individual
Yes
Use crossover to generate an indivudual not in the tabu list and select it for local seach
No
Select the individual from the population and select it for local seach
Add the individual into the gene pool
Do local search to find the local optimal
Resize the population and tabu list to meet the upper bounds
Meet the stop condition?
Add the origin selected individual and local optimal into the tabu list
Remove the individual from the populations if exists
Output the solution

Fig. 1: General workflows

$|E| = O(|V|^2)$, the time complexity reach to $O(|V|^3)$ or even higher depending on heap implementation. Moreover, the Implementation of Floyd-Warshall Algorithm is much simpler with small time constant. Thus, we selected Floyd-Warshall Algorithm for graph preprocessing steps.

2) Population Initialization: The population initialization steps can significantly affect the final solution of the Evolutionary Algorithm. Thus,we primarily utilize the Path Scanning Algorithm [13] (PS), which is a greedy based approach that can generate high-quality individuals. The Path Scanning Algorithm mainly include three steps: First, it select all the edges to determine the candidate served edges that have minimum distance from the edge of the current route and not result in the exceed of the capacity limitation. Second, it select one edge from the candidates based on the rules. Third, it insert the edge into the route until the route reach the capacity limitation or no edge is not served. The pseudo-code of the steps is shown in Algorithm 1.

For the rules to select single edge from candidate edges, MAENS have proposed several strategy for edge selection [21]. 1) maximize the distance from the head of task to the depot; 2) minimize the distance from the head of task to the depot; 3) maximize the term $es.d/sc(es)$, where $sc(es)$ is the serving cost of edge es; 4) minimize the term $es.d/sc(es)$; 5) use rule 1) if the vehicle is less than half full, otherwise use rule 2). In addition, we also utilize the strategy 6) that randomly select one edge from the candidate.

Based on the Path Scanning Algorithm, we can further developed our Population Initialization Algorithm. Given a

Algorithm 1 Path Scanning

Input: Demand Edges Sets T, min distance matrix M, capacity limitation Q, selecting rule RE
Output: A valid set of routes R

```
1:  $R \leftarrow []$
2:  while Size($T$) $> 0$ do
3:      $r \leftarrow Path()$
4:      while True do
5:          Set the $node_{prev}$ as the end vertex of current route if the route is not empty, otherwise set as the initial depot
6:          $candidate \leftarrow []$,$d_{min} \leftarrow \infty$
7:          for $e \in T$ do
8:              if $e.demand + r.load < Q$ then
9:                  Get the tuple sets $tups$ in two direction i.e. $(v, w), (w, v)$ for undirected edge $e = (v, w)$
10:                 for $tup \in tups$ do
11:                     $d_{cur} \leftarrow M[node_{prev}][tup.from]$
12:                     if $d_{cur} < d_{min}$ then
13:                         $d_{min} \leftarrow d_{cur}$
14:                         $candidate \leftarrow [tup]$
15:                     else if $d_{cur} =$ then
16:                         $candidate$.UpdateByRule($tup$,$RE$)
17:                     end if
18:                 end for
19:             end if
20:         end for
21:         if Size($candidate$) $= 0$ then
22:             break
23:         end if
24:         $selected \leftarrow$ RandomSelection($candidate$)
25:         Appending $tup$ into the $r$, with the cost and load updated correspondingly
26:     end while
27:     $R$.Append($r$)
28: end while
```

pre-processed graph with additional information, the algorithm iteratively runs Path Scanning Algorithm (with each invoke given a candidate selection rule), to generate $psize$ different individuals that not in tabu list. If the PS cannot generate new individual in several attempts, the algorithm terminates if $psize_{cur} > 0$, otherwise another invocation starts on Population Initialization with Amnesty Rule enabled. By utilizing the Amnesty Concepts [12], we given our definition of Amnesty Rule. When Amnesty enabled, if individual generated is in tabu list, the algorithm remove the individual in the tabu list. This allows the algorithm to continue to run.

3) Individual Selection and Mutation: The Individual Selection and Mutation Algorithm mainly utilize the thought of the Genetic Algorithm. Initially, we randomly select one individual from the population as s_1 and another from the gene pool as s_2. We then chose whether or not to crossover the two individuals based on a certain probability and the gene pool size (check whether the gene pool size is larger than expected). The output of this stage is an individual ind_{origin} that not in the tabu list and will continue processed in local search steps shown in III-B4.

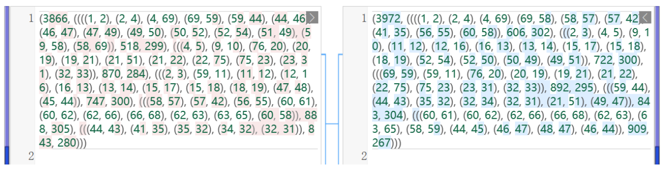
The crucial steps is the implementation of the crossover steps, which we mainly adopt the approaches proposed in MAENS [21]. First, it select the route r_1 from the s_1 and r_2 from the s_2, which is then split randomly into sub-sequences r_{11},r_{12},r_{21},r_{22}. Then we combine the r_{11} and r_{22} as a replacement of the r_1 in the s_1, where we remove the duplication edge and recover the missing edges by re-inserted them to the near-optimal position, as pseudo-code shown below:

Algorithm 2 Crossover

Input: Individual s_1, s_2, tabu list $tabu$, remove mode $rmv \in \{True, False\}$
Output: A individual s_x not in tabu list

```
1:  while attempts under the limitation do
2:      $r_1 \leftarrow$ RndSelectRoute($s_1$),$r_2 \leftarrow$ RndSelectRoute($s_2$)
3:      $r_{11}, r_{12} \leftarrow$ Seg($r_1$),$r_{21}, r_{22} \leftarrow$ Seg($r_2$)
4:      $missing \leftarrow$ Set($r_{12}$)
5:      for $tup \in r_2$ do
6:          if $tup \in missing$ then
7:              Append($r_1 1$,$tup$)
8:          end if
9:      end for
10:     for $tup \in missing$ do
11:         Insert to $r_{11}$ with the min increment of cost
12:     end for
13:     $s_x$ = ReplaceR1ByNew(copyOf($s_1$), $r_1, r_{11}$)
14:     if $sx \notin tabu$ then
15:         break
16:     end if
17: end while
18: if Failed to generate individual then
19:     Crossover($s_1$,$s_2$,$tabu$,$rmv = True$)
20: end if
```

After the local search is conducted, we use the output individual ind_{ls} to update the best found individual by checking whether it is better than it. The algorithm further update the gene pool and the tabu list by adding the individuals ind_{origin} and ind_{ls} into the gene pool and tabu list. This is due to the observation that the ind_{ls} might contains several optimal subsequences (sequences of tasks that in optimal or near-optimal order), which is considered as "good gene" that can be used in crossover. The example of two different ind_{ls} comparison shown in fig.2 demonstrates the similar sequence like *(60, 61), (60, 62), (62, 66), (66, 68), (62, 63), (63, 65)* which did not exist before the local search, which is considered as *near-optimal sequences*



```
(3866, ((((1, 2), (2, 4), (4, 69), (69, 59), (59, 44), (44, 46
(46, 47), (47, 49), (49, 50), (50, 52), (52, 54), (51, 49), (5
9, 58), (58, 69)), 518, 299), (((4, 5), (9, 10), (76, 20), (20,
19), (19, 21), (21, 51), (21, 22), (22, 75), (75, 23), (23, 3
1), (32, 33)), 870, 284), (((2, 3), (59, 11), (11, 12), (12, 1
6), (16, 13), (13, 14), (15, 17), (15, 18), (18, 19), (47, 48),
(45, 44)), 747, 300), (((58, 57), (57, 42), (56, 55), (60, 61),
(60, 62), (62, 66), (66, 68), (62, 63), (63, 65), (60, 58)), 88
8, 305), (((44, 43), (41, 35), (35, 32), (34, 32), (32, 31)), 8
43, 280)))
```

```
(3972, ((((1, 2), (2, 4), (4, 69), (69, 58), (58, 57), (57, 42
(41, 35), (56, 55), (60, 58)), 606, 302), (((2, 3), (4, 5), (9, 1
0), (11, 12), (12, 16), (16, 13), (13, 14), (15, 17), (15, 18),
(18, 19), (52, 54), (52, 50), (50, 49), (49, 51)), 722, 300),
(((69, 59), (59, 11), (76, 20), (20, 19), (19, 21), (21, 22),
(22, 75), (75, 23), (23, 31), (32, 33)), 892, 295), (((59, 44),
(44, 43), (35, 32), (32, 34), (32, 31), (21, 51), (49, 47)), 84
3, 304), (((60, 61), (60, 62), (62, 66), (66, 68), (62, 63), (6
3, 65), (58, 59), (44, 45), (46, 47), (48, 47), (46, 44)), 909,
267)))
```

Fig. 2: Optimal Similarity Examples

Also to improve the diversity of the gene pool, ind_{origin} is added into the gene pool. For the updating of the tabu list, since ind_{ls} is considered as a local optimal that cannot be improved by local search, and since the local search is less stochastic steps, resulting that different call on ind_{origin} likely obtain the similar ind_{ls}, we also add the individual into the tabu list and apply the Amnesty Rule only after a full search for other initial individuals.

4) Local Search: The local search steps mainly adopt the idea of Hill Climbing Algorithm to find the local optimal solutions. After evaluating the effectiveness of the operators shown in MAENS [21], the algorithm adopted 2 operators that performs best: Merge-Split Operator(MS) and Single Insertion Operator(SI). The decision that not use the other operators is due to that other operators might be less likely to produce a better result than selected operators considering their time costs.

For the Single Insertion Operator, the operator select every required edge in the route, and for each selected edge, it is removed from the origin place and re-inserted to the optimal place that can minimize the total cost.

For the Merge-Split Operator, we modify the operating process proposed in MAENS to satisfy the tabu search. The Merge-Split Operator runs $msTimes$ iterations, and for each iteration, it randomly select two routes, merging the demand edge in the route and reuse the Path Scanning Algorithm described in III-B2 to find a better arrangement not in tabu list, and put the output routes back to the solutions. The Merge-Split will also terminates early if no different arrangement found after times of attempt.

This local search steps is less stochastic, since the Single Insertion Operator is deterministic and the Merge-Split Operator is also near-deterministic due to that for large $msTimes$ to run the iteration, the operator is likely to produce same near-optimal arrangement.

C. Performance Analysis

1) Complexity Analysis: In this section, we analyze the time complexity of the important algorithm and operators mentioned in III-B, including Path Scanning Algorithm, Crossover Algorithm and two Operators, which mainly determine the speed of each iteration.

Path Scanning Algorithm: There are $O(|T|)$ slots need to be filled with the required edges, where $|T|$ represents the number of tasks, and for each slot, we scanning all the possible edge to find the best candidate, taking $O(|T|)$, thus the total time complexity is $O(|T|^2)$

Crossover Algorithm: The first step to put the edges in r_{22} to r_{11} takes $O(|T|)$, and the consequent recover step involves $O(T|)$ edge to recover, with each takes $O(|T|)$ space to determine the best position, thus the total time complexity is $O(|T|^2 + |T|) = O(|T|^2)$

SI Operator: This Operator select all the edge with $O(|T|)$ amount for re-insertion attempt. With $O(|T|)$ scan for each attempt to find the optimal insertion, thus the total time complexity is $O(|T|^2)$

MS Operator: This Operator attempts k times merge-split operation, for each operation, a path scanning is conducted, thus the total time complexity is $O(k|T|^2)$

By the above analysis, we observed that the total complexity of the algorithm is approximately $O(|T|^2)$, which is in acceptable speed, significantly helps in improving the convergence speed of the algorithm, as demostrated in IV

2) Optimality: Utilizing randomness and statistical simulation, the TGA does not ensure optimal decision-making but rather aims to discover a satisfactory solution within a reasonable time. Nonetheless, two crucial factors influence the quality of the solution:

Iteration Time: The number of iteration time significantly impacts the final evaluation outcomes, while more iteration can increase the possibility to provide a better solution.

Diversity Of Gene Pool: The diversity of the gene pool determine the possibility to generate promising individuals. If the gene pool is large and diverse enough, it is more likely to contain high-quality genes.

IV. Experiments

In this section, we aim to evaluate the performance and the convergence rate of the TGA. Our experiments will primarily focus on: (a) The performance of TGA comparing with the SOTA algorithm MAENS [21] (b) The effectiveness of combining the thought of Tabu Search, Genetic Search and Hill Climbing Algorithm.

A. Setup

Configuration:
CPU: 12th Gen Intel(R) Core(TM) i7-12700H 2.30 GHz
RAM: 16G
Operating System: Windows 11 23H2
Python Version: 3.9.7

B. Results And Analysis

1) Effectiveness evaluation on TGA: We set the baseline as MEANS to evaluate the performance of our proposed algorithm. We will run the TGA by 10 times in 60s on each instance of the provided dataset and part of additional large dataset [7], [8], [16], then comparing the average and best solution provided with the best and average provided by MAENS. To evaluate the performance differences among different instance in dataset, we define the DIF that normalize the difference as below:

$$DIF = (Avg_{TGA} - Best_{MAENS})/Best_{MAENS}$$

The result of the experiment is shown as below, where T is in short of TSA and M is in short of MAENS

Table 1: Result of Effectiveness Evaluation of TGA

The above table demonstrate the efficiency of the TSA. For the smaller dataset gdb and val series, the TSA can steadily output the optimal solution. And for larger dataset egl series, the TSA can also output a solution that with tiny differences ($DIF \leq$ 0.5) to MAENS, showing the robustness of the

Instance	Avg_T	Std_T	$Best_T$	Avg_M	$Best_M$	DIF
gdb1	316	0.00	316	316	316	0.00
gdb10	275	0.00	275	275	275	0.00
val1A	173	0.00	173	173	173	0.00
val4A	400	0.00	400	400	400	0.00
val7A	279	0.00	279	279	279	0.00
egl-e1-A	3608	0.00	3608	3548	3548	0.01
egl-s1-A	5234.1	11.33	5228	5039.8	5018	0.04
egl-s2-C	17255.0	63.7	17210	16509.8	16430	0.05
egl-s3-C	17979.6	47.49	17934	17305.8	17207	0.04
egl-s4-C	21582.6	23.95	21567	20767.2	20538	0.05

algorithm. Also, the standard deviation of the algorithm is minor, demonstrating the stability for the algorithm to produce good result.

2) Effectiveness of the combination of GA,TS and HCA: We performed ablation experiments on various key parts of the TGA, through comparing TGA with the GA that remove Tabu from TGA, TS that remove the gene pool and TG that remove the Local Search, to evaluate the effectiveness of the combination of GA, TS and HCA. We restrict the four algorithms to run 60s on instance val4A and egl-s4-C, the following figure shows the curve of cost over time for the best solution:

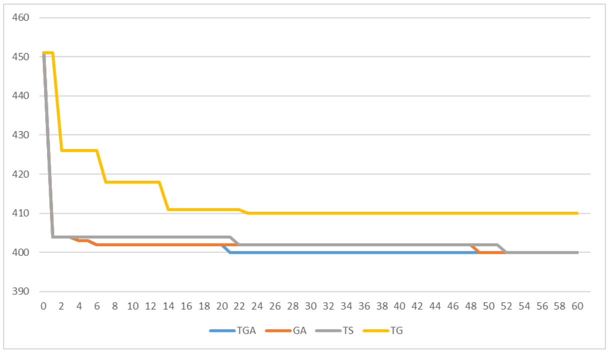


Fig. 3: Cost of best solution over time on dataset val4A

As shown in graphs, with the same initial population, the local search algorithm significantly improved the convergence speed and the final output of the total algorithm, where algorithm that have local search can generate high quality individual from the initial ones. Also, The convergence speed of the TGA is the fastest among the algorithm, which is easier to jump out of the local optimal(like cost=404,402 in 3, where TGA has the shortest dwell time at this value). In comparison, the GA that without the tabu list is easier to spend unnecessary time testing similar individual, while the TS without crossover is far less diversity that makes it hard to jump out of the local optimal. Our experiment demonstrate the effectiveness of our combination.

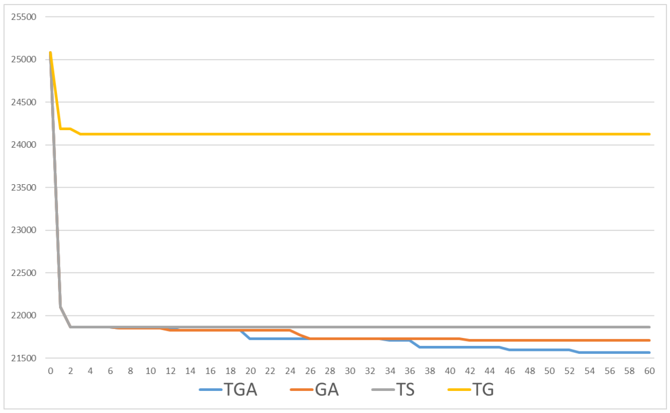


Fig. 4: Cost of best solution over time on dataset egl-s4-C

V. Conclusion

In this paper, I propose a hybrid evolutionary algorithm proposed in paper named as TGA that combines the Genetic Algorithm (GA), Tabu Searching Algorithm(TS) and Hill Climbing Algorithm(HCA) for solving the CARP. I also analyze the overall complexity and performance of the model. Additionally, I conduct empirical studies evaluate the performance of our algorithm. The experimental results show that the model can output the solution that is close to SOTA $(DIF \leq 0.5)$ in limited time. We also demonstrate that the combination of GA,TS and HCA effectively boost the convergence speed and result quality of the algorithm, allowing our algorithm to quickly converge to a superior solution.

Due to time limitation, the algorithm did not use other operators like swap, 2-opt [21], which though may reduce the speed of convergence due to their time taken, these operators may improve the diversity of the gene pool and thus output a better solution. At the same time, this algorithm can further consider accepting illegal individuals and adjusting them to legal ones, which is likely to increase the diversity and reduce the difficulty to jump out of local optimal [19], improving the overall performance.

In this project, I delved into evolutionary algorithms and finally proposed an effective hybrid algorithm through comparative attempts, which significantly improved my understanding and knowledge of AI. Meanwhile, I enhanced my understanding of python and programming, such as the design of multi-process program. These will benefited me a lot in further study.

VI. Acknowledgement

I am deeply grateful to Professor Bo Yuan for his outstanding course, which has enriched my understanding of Evolutionary Algorithms. Furthermore, I extend my heartfelt appreciation to the dedicated SAs for their invaluable guidance during lab sessions. The opportunity to delve into the intricacies of CS311 and apply these concepts to our project has been immensely rewarding. I apologize for any inconvenience caused by the lengthy reference list.

References

[1] E. Benavent, V. Campos, A. Corberan, and E. Mota. The capacitated arc routing problem: Lower bounds. *Networks*, 22(7):669–690, 1992.

[2] W. Chenge and C. Ada. Review of capacitated arc routing problems. 2022.

[3] Corberán, R. Eglese, G. Hasle, I. Plana, and J. M. Sanchis. Arc routing problems: A review of the past, present, and future. *Networks*, 77, 06 2020.

[4] Corberán and G. Laporte. *Arc Routing: Problems, Methods, and Applications*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 2015.

[5] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[6] M. Dror. *Arc Routing: Theory, Solutions and Applications*. 01 2000.

[7] R. W. Eglese. Routing winter gritting vehicles. *Discrete Appl. Math.*, 48(3):231–244, feb 1994.

[8] R. W. Eglese and L. Y. O. Li. A tabu search based heuristic for arc routing with a capacity constraint and time deadline. 1996.

[9] S. Elatar, K. Abouelmehdi, and M. E. Riffi. The vehicle routing problem in the last decade: variants, taxonomy and metaheuristics. *Procedia Computer Science*, 220:398–404, 2023. The 14th International Conference on Ambient Systems, Networks and Technologies Networks (ANT) and The 6th International Conference on Emerging Data and Industry 4.0 (EDI40).

[10] R. W. Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, jun 1962.

[11] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, jul 1987.

[12] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers Operations Research*, 13(5):533–549, 1986. Applications of Integer Programming.

[13] B. Golden, J. Dearmon, and E. Baker. Computational experiments with algorithms for a class of routing problems. *Computers Operations Research*, 10(1):47–59, 1983.

[14] B. L. Golden and R. T. Wong. Capacitated arc routing problems. *Networks*, 11(3):305–315, 1981.

[15] C. Hess, A. G. Dragomir, K. F. Doerner, and D. Vigo. Waste collection routing: a survey on problems and methods. *Central European Journal of Operations Research*, 32(2):399–434, 2024.

[16] L. Y. O. Li and R. W. Eglese. An interactive algorithm for vehicle routeing for winter — gritting. *Journal of the Operational Research Society*, 47(2):217–228, 02 1996.

[17] M. Liu and T. Ray. Efficient solution of capacitated arc routing problems with a limited computational budget. In M. Thielscher and D. Zhang, editors, *AI 2012: Advances in Artificial Intelligence*, pages 791–802, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[18] A. Madkour, W. G. Aref, F. ur Rehman, M. A. Rahman, and S. M. Basalamah. A survey of shortest-path algorithms. *ArXiv*, abs/1705.02044, 2017.

[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)*, 39(3):723–734, 2009.

[20] W. Pearn. New lower bounds for the capacitated arc routing problem. *Networks*, 18:181 – 191, 10 2006.

[21] K. Tang, Y. Mei, and X. Yao. Memetic algorithm with extended neighborhood search for capacitated arc routing problems. *IEEE Transactions on Evolutionary Computation*, 13(5):1151–1166, 2009.

[22] K. Vaishali, M. Lakra, R. Sahni, and R. Verma. Shortest path algorithms: Comparison and applications. *A Multidisciplinary Research Journal*, page 28, 2022.