Int J Syst Assur Eng Manag (December 2023) 14(6):2684–2694
https://doi.org/10.1007/s13198-023-02161-2

ORIGINAL ARTICLE

A comparative analysis of genetic algorithms on a case study of asymmetric traveling salesman problem

Amit Raj[1] · Parul Punia[1] · Pawan Kumar[1]

Received: 8 April 2023 / Revised: 16 August 2023 / Accepted: 7 September 2023 / Published online: 7 October 2023

Abstract In the present paper, the genetic algorithm and some of its variants i.e. adaptive genetic algorithm, binary-coded genetic algorithm and real-coded genetic algorithm are applied to the Asymmetric Traveling Salesman Problem (ATSP). ATSP is one of the most widely studied combinatorial NP-hard problems of finding the shortest path. The present ATSP is a novel real-life case of the shortest path problem based on the distances between 22 districts of Haryana, India. To solve the above problem, one-point crossover and exchange mutation are applied to compare the performance of these algorithms on different parameters such as the size of the population, the number of iterations, and the rate of crossover. The main objective of this paper is to study the influence of these parameters on ATSP. Numerical results show that the binary genetic algorithm worked better in terms of the size of the population and the number of iterations, while the real-coded genetic algorithm worked better in terms of the rate of crossover.

Keywords Asymmetric traveling salesman problem · Genetic algorithms · Population size · Crossover

1 Introduction

The Traveling Salesman Problem (TSP) is a classical optimization problem introduced by Bellmore and Nemhauser (1968) that has gained recognition in the fields of graph theory and operations research. This classic problem entails finding the shortest route for a salesman to visit a set of cities exactly once before returning to the starting point. However, as the number of cities increases, the complexity of the TSP grows exponentially, rendering the search for optimal solutions computationally inefficient and known for its complexity as it falls into the NP-hard class (Gary and Johnson 1979).

Despite its notorious difficulty and classification as NP-hard, TSP is one of the most widely studied problems in computational mathematics across various domains including vehicle routing (Adewumi and Adeleke 2018), computer wiring, engineering design (Li et al. 2022), machine sequencing (Raj and Bhattacharyya 2018; Mahapatra et al. 2021; Dehedkar and Raj 2022; Mahapatra and Raj 2023; Raj et al. 2023) and scheduling (Bansal and Singh 2022), as well as frequency assignment in communication networks.

Based on the structure of the distance matrix, the TSP problem is divided into two types: symmetric and asymmetric. The distance between any two cities is the same, regardless of the order in which they are visited is known as Symmetric Traveling Salesman Problem (STSP) while the distance between two cities may differ depending on the direction of travel is known as Asymmetric Traveling Salesman Problem (ATSP). This study is focused on solving ATSP. Mathematically, ATSP is represented as $d(i,j) \neq d(j,i)$, where d(i, j) represents the distance from city i to city j, and d(j, i) represents the distance from city j to city i.

The primary objective of the ATSP is to find the most efficient tour that allows a traveling salesman to visit each city exactly once and return to the starting point, considering the asymmetrical distances. There are some exact methods such as branch and cut (Ascheuer et al. 2000), etc in the literature for solving ATSP problems, but all of them have

✉ Pawan Kumar
drpawan@cuh.ac.in; chhoker.pawan@rediffmail.com

[1] Department of Mathematics, Central University of Haryana, Jant-Pali, Mahendergarh, Haryana 123031, India

exponential complexity, need too much computation time, or use too much memory to obtain the optimal solution.

In contrast to exact algorithms, new metaheuristic approaches are being utilized to address the NP-hard ATSP. Some of these approaches include the genetic algorithm (Nagata and Soler 2012), harmony search algorithm (Boryczka and Szwarc 2019), discrete bat algorithm (Osaba et al. 2016), discrete water cycle algorithm (Osaba et al. 2018) and discrete mayfly algorithm (Zhang et al. 2023).

Even though these metaheuristic algorithms have been applied to the ATSP, there is a gap in the existing literature when it comes to comparing various variants of the genetic algorithm (GA) against the standard GA on real-life instances of the ATSP. This research gap highlights the need for a comprehensive study that systematically evaluates and compares the performance of various GA variants on a real-life ATSP.

Genetic algorithms (Goldberg 1989; Deepa 2008; Katoch et al. 2021) belong to the larger class of evolutionary algorithms inspired by biological evolution. Holand's GA (1970) is a powerful optimization technique that generates solutions to optimization problems using natural selection and genetics techniques. These algorithms are typically quite simple and have a quick processing time. As a result, using genetic algorithms and their variants to solve an NP-hard problem such as ATSP may be appropriate. That is why, this study is focused on evaluating the performance of simple GA and its variants i.e., adaptive genetic algorithm (AGA), binary-coded genetic algorithm (BGA), and real-coded genetic algorithm (RCGA). This evaluation is carried out on a real-life ATSP problem involving 22 districts of Haryana, India.

The main contributions of this research are as follows:

- A novel real-life ATSP problem is formulated by using 22 districts of Haryana, India.
- Some Variants of the GA algorithm have been applied to the formulated ATSP problem in MATLAB.
- one-point crossover and exchange mutation are two genetic operators used in this study.
- A comparative analysis has been carried out to evaluate the performance of GAs on different parameters such as the size of the population, the number of iterations, and the rate of crossover.
- The evaluated results show that the binary genetic algorithm worked better in terms of the size of the population and the number of iterations, while the real-coded genetic algorithm worked better in terms of the rate of crossover.

The subsequent sections of the paper are structured as follows: Sect. 2 explains the related work on the ATSP problem. Section 3 introduces the real-life ATSP problem. Section 4 presents the mechanism of the genetic algorithms. Section 5 presents the genetic operators of GAs. Section 6 presents the implementation of GAs on formulated ATSP. Section 7 contains the results and discussion. Finally, Sect. 8 presents the conclusion of the paper.

2 Related work

Over the past fifty years, researchers have developed numerous algorithms, both exact and heuristic, to solve TSP. Balas and Christofides (1981) came up with a new approach called the restricted Lagrangean algorithm to solve TSP by adding linear inequalities to the constraints in a clever way. Deep et al. (2018) modeled a traveling salesman problem involving seven cities connected by Indian railways. To solve this problem, a genetic algorithm (GA) is employed with the fourth variant of order crossover (OX4) as proposed in Deep and Mebrahtu's work and two mutation operators, namely inversion mutation and inverted displacement mutations, were incorporated.

For more surveys on solution methods for the TSP, the reader may refer to Fiechter (1994), Carpaneto et al. (1995), Potvin (1996), and Larranaga et al. (1999). We also strongly recommend Tawhid and Savsani (2019), Akhand et al. (2020), Li et al. (2023), Rocha and Subramanian (2023), and Mzili et al. (2023). These algorithms encompass iterative improvement methods, construction procedures, branch-and-bound exact algorithms, as well as popular meta-heuristic approaches like Ant Colony (AC), Genetic Algorithm (GA), Tabu Search (TS), Sine-Cosine Algorithm and Artificial rat optimization algorithm.

Exact and metaheuristic algorithms have been proposed for both symmetric TSP and asymmetric TSP cases but this study is focusing on the ATSP problem. Pekny and Miller (1990) introduced a parallel branch and bound algorithm to solve the ATSP problem. The algorithm employs various techniques, including an assignment problem-based lower bounding technique, subtour elimination branching rules, and a subtour patching algorithm for upper bounding. Ascheuer et al. (2000) introduced a branch and cut algorithm for ATSP with precedence constraints. Apart from these exact algorithms, researchers have also successfully applied meta-heuristic methods to solve the ATSP.

Buriol et al. (2004) proposed a new memetic algorithm designed specifically to solve ATSP. The algorithm includes a new and effective local search called the recursive arc insertion (RAI) mechanism along with various innovative features. These features encompass a complete ternary tree structure with thirteen nodes to organize the population topologically, a hierarchical organization of overlapping clusters leading to a unique selection scheme, and the implementation of efficient data structures.

Majumdar and Bhunia (2011) introduced a novel GA approach to address a realistic version of the ATSP focused on time minimization. The problem incorporates

inter-city travel times represented as intervals and the approach is designed by combining local GA (LGA) and Global GA (GGA). GGA is employed to search for global optima within the main tours, while the LGA is specifically applied to randomly selected sub-tours derived from the main tour obtained by GGA. This use of LGA allows for the exploration of local optimal solutions within the sub-tours, contributing to improved overall performance. Nagata and Soler (2012) came up with a new operator called the edge assembly crossover (EAX) operator to modify GA and applied it to an ATSP problem.

Osaba et al. (2018) introduced a discrete version of the Water Cycle Algorithm (DWCA) tailored for efficiently solving two well-known optimization problems: TSP and ATSP. DWCA retains its inspiration from hydrological phenomena but incorporates novel elements to address these routing problems effectively. The algorithm uses the Hamming distance to measure differences between routes found during the search process, adapts the movement function based on the estimated inclination of the river, and employs an insertion-based mutation operator that emulates evaporation and raining processes in the discrete solution space encoded by permutations.

Zhang et al. (2023) proposed a Discrete Mayfly Algorithm (DMA), a swarm-based metaheuristic, specifically designed for the spherical ATSP. The DMA utilizes various operators, including inver-over, crossover, and 3-opt, to simplify parameters, enhance population diversity, and improve local search capabilities.

3 Problem statement

The TSP has numerous real-world applications across various domains, including logistics, transportation, manufacturing, and network routing. In road networks, it is common for the travel distance between two locations to be different in opposite directions due to factors like road conditions, traffic flow, and road restrictions. This asymmetry in travel distances creates a more complex and challenging problem, making it essential to explore specialized algorithms to address such real-life scenarios. In this specific case, solving the ATSP for the 22 districts of Haryana, India has different distances between nodes in both directions, which is of great importance in various scenarios, such as logistics, transportation planning, and resource allocation.

The state of Haryana, with its 22 districts, represents a significant geographical area with various important locations that require efficient visitation by a traveling salesman. By formulating this problem as an ATSP and using real-world data extracted from Google Maps, we can model and evaluate the most efficient routes for the salesman to visit all the districts while minimizing the overall travel distance. This analysis can lead to insights and decision-making tools that can benefit businesses, government agencies, and other organizations operating in Haryana. With this context in mind, 22 districts of Haryana, India modeled as an ATSP problem in the study.

Each district is represented by a node and is numbered from 1 to 22. Also, the shortest distance between node 1 to node 2 is not always the same as the distance between node 2 to node 1. In matrix notation of this ATSP, the entries above the diagonal represents the maximum distance between the nodes and the entries below the diagonal represents the minimum distance between them. Faridabad, Gurugram, Panchkula, Karnal, Hisar, Panipat, Ambala, Kurukshetra, Rohtak, Jind, Rewari, Jhajjar, Kaithal, Bhiwani, Fatehabad, Sirsa, Yamunanagar, Sonipat, Palwal, Mahendergarh, Charkhi Dadri, and Nuh are the 22 districts of Haryana represented by nodes 1 to 22 respectively. The data used in the present study is taken from Google Maps and is presented in Table 1. By exploring this formulated problem, we aim to compare different GA variants and assess their performance in finding the most efficient routes for a salesman to visit all the districts in Haryana, India while minimizing the overall travel distance.

4 Genetic algorithms

In this section, we discuss genetic algorithm (or simple genetic algorithm) and some of its variants, i.e., adaptive genetic algorithm, binary-coded genetic algorithm, and real-coded genetic algorithm. Since GAs are the most popular algorithms, we just introduce the basic working principles of these algorithms.

4.1 Simple genetic algorithm (SGA)

The mechanism of simple genetic algorithm is very simple, involving copying and exchanging bits of partial strings. Reproduction, crossover, and mutation are three operations in the algorithm. Depending on the fitness value of the chromosome, a reproduction operator is applied to generate parent chromosomes or strings. After that, offspring are produced by applying a crossover operator to the parent chromosomes and then a mutation operator is used to keep the population diverse. A Pseudo Code is used to summarise the steps involved in the process of SGA is given below:

Table 1 The formulated asymmetric travelling salesman problem is in matrix form

Cities	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
1	0	45	295	169	239	136	296	208	119	201	98	88	224	157	290	310	264	92	36	148	135	81
2	39	0	315	190	176	152	273	228	81	162	68	47	209	122	240	268	269	112	82	113	95	66
3	287	293	0	123	247	194	49	97	257	205	340	298	126	283	241	282	96	255	358	330	294	359
4	160	157	123	0	173	46	81	36	145	103	211	158	79	163	179	224	77	113	257	227	202	234
5	208	166	241	154	0	176	210	185	113	74	185	132	127	81	64	98	239	166	243	136	110	223
6	128	136	157	35	143	0	150	95	94	91	175	123	97	134	182	224	113	74	214	190	156	196
7	249	253	42	81	199	117	0	50	216	139	297	246	87	245	205	240	81	196	334	310	252	353
8	198	209	90	35	167	72	50	0	162	114	248	195	69	199	195	222	53	158	287	265	239	306
9	111	79	236	114	101	76	196	150	0	83	100	45	133	76	168	210	210	50	150	110	66	123
10	164	155	180	83	71	71	139	102	67	0	173	120	74	91	122	149	158	107	229	170	102	206
11	89	54	327	203	158	165	287	240	89	161	0	58	227	99	234	278	277	121	93	55	70	51
12	84	44	278	153	120	119	236	187	36	108	52	0	174	78	181	225	223	85	116	81	43	92
13	216	200	121	62	116	81	80	51	118	56	215	161	0	140	133	156	120	139	269	208	169	252
14	151	110	248	151	62	113	207	182	50	69	97	65	125	0	132	182	228	108	185	68	29	168
15	260	218	216	170	50	167	197	165	151	98	226	172	114	115	0	41	237	189	290	189	157	274
16	306	264	281	212	93	214	234	206	193	141	274	217	156	156	41	0	258	259	334	225	207	316
17	232	237	92	64	219	100	60	45	179	146	274	218	99	217	216	258	0	178	306	285	250	300
18	87	86	206	81	149	49	164	117	47	83	114	61	127	104	182	228	146	0	155	155	115	129
19	31	54	353	196	226	197	278	233	139	217	80	100	262	169	276	322	295	146	0	148	151	36
20	144	107	316	207	129	169	274	244	96	136	55	65	193	68	182	225	272	135	133	0	40	132
21	129	87	278	166	90	129	236	207	56	98	68	43	154	29	139	186	232	99	145	40	0	129
22	59	47	323	212	208	180	295	253	122	198	47	81	244	150	257	303	282	129	36	104	117	0

Algorithm 1 Simple Genetic Algorithm (SGA)

1: Initialize the parameters such as Population size, crossover probability, and mutation probability.
2: Generate an initial population
3: Calculate the objective function (or fitness) of the chromosomes
4: Choose the individuals with higher fitness values to have a greater chance of surviving in the next generation (Selection Process)
5: Perform Crossover operator
6: Perform Mutation operator
7: Calculate the fitness value
8: Update best candidate solution
9: Output best candidate solution obtained by SGA

4.2 Binary coded genetic algorithm (BGA)

Depending upon the representation of the chromosomes genetic algorithm can be classified into binary genetic algorithm and real coded genetic algorithm. The chromosomes in binary coded genetic algorithm are encoded in the string of bits 0 and 1. The length of bit string is kept fixed. Then suitable crossover and mutation operators are applied on the population. To perform mutation random number for each bit of the string is generated and its value are swapped 0 for 1 and vice-versa. After obtaining the solutions they are then again converted into actual values. For more details about the algorithm, go through the paper (Kim et al. 2002; Mohebifar 2006).

4.3 Real coded genetic algorithm (RCGA)

Real coded genetic algorithm (Goldberg 1991) has strings made of real numbers unlike those of binary coded genetic algorithm and also takes the operators reproduction, crossover and mutation of real vectors. Hence, the representation of the chromosomes is very close to natural formulation of many problems and by the use of real parameters large and continuous domains can be easily searched as RCGA works well with the continuous space. The chromosomes in RCGA are bounded depending on the variables they represent. For more details about the algorithm, go through the paper (Eshelman and Schaffer 1993; Singh et al. 2015; Ali et al. 2018; Wang et al. 2019).

4.4 Adaptive genetic algorithm (AGA)

The basic idea behind adaptive genetic algorithm (Lin 2009) was to prevent pre-mature convergence of the algorithm arising due to lack of diversity in the population and unbalanced exploration and exploitation rate. In AGA, the parameters like population size, crossover and mutation rate are varied while the GA is still running so as to maintain diversity and exploitation and exploration rate in the population. For more details about the algorithm, go through the paper (Wang et al. 2008; Saptarini et al. 2020).

5 Genetic operators

In this section, we discuss the essential genetic operators used in genetic algorithms: encoding, selection, crossover, and mutation. Each of these operators plays a crucial role in shaping the algorithm's efficiency, convergence, and exploration of the solution space. By applying these operators intelligently, we aim to find the most efficient routes for a traveling salesman to visit all the districts, thereby minimizing the overall travel distance and optimizing the solution.

5.1 Encoding or design the chromosome

A population is an accumulation of chromosomes. Each chromosome represents a possible solution and contains several genomes. An algorithm's efficiency and processing power rely on the encoding of a problem. Designing a chromosome to represent a problem is a significant architectural decision in the algorithm. So, the primary focus is to design the chromosomes so that they can have a considerable influence on processing speed, convergence, and the overall ease of crossover and mutation. In addition to being the most effective and efficient method for encoding parameters into chromosomes, the representation's underlying shape is also crucial. It is also important that the values of a chromosome are a solution's parameters, and they must always be accompanied by an evaluation process that uses the parameters to determine the system's aspect or conclusion. Given the value of the chromosome's parameters, the fitness function measures how much this outcome is a better or worse solution.

5.2 Selection operator

Selection operator chooses the best possible chromosomes from the existing population to create the next generation. In various situations, a distinct selection procedure is utilized for the purpose of picking the population of the healthiest individuals, parents, or chromosomes for crossover or mutation.

5.3 Crossover

Crossover is the process of making new solutions from solutions that already exist. It is a form of reproduction through sexual means. To generate superior offspring, a random selection is made from the mating pool to choose two different string combinations to crossover. The approach that is selected is determined by the encoding method.

5.4 Mutation

Mutation is a small random change in the chromosome to produce a new solution. Crossover breeding can often provide too much variability, making it difficult to fully explore the underlying solution space. To maintain the diversity in the population, a mutation operator is used.

6 Implementation

In Sect. 4, the basic ideas of genetic algorithms were introduced. In this section, we will discuss the actual working & the time and space complexity of various genetic algorithms to solve the formulated ATSP problem.

6.1 Basic steps of algorithms

The various steps of implementation of genetic algorithms to our real-life problem are as follows:

Generate the initial population Generate a random initial population of potential solutions, where each chromosome represents a valid tour visiting all 22 districts exactly once. Each chromosome is a permutation of the numbers from 1 to 22, representing the order of city visits.

Fitness Value Define the fitness function to calculate the total distance of each tour in the population. The fitness value of each individual (chromosome) is the sum of the distances between consecutive cities based on the matrix notation. The fitness value represents the length of the tour, and it should be used to compute the fitness score for selection. The fitness value can be calculated using the formula,

$$F_{Individual} = \sum_{i,j=1}^{22} a_{ij} \tag{1}$$

where a_{ij} is the distance between the two adjacent cities i and j of the tour.

Selection Operator The roulette wheel selection operator is used in this work for selecting the parent chromosomes. The value of the fitness function and the corresponding probability of every individual is calculated. Each individual is assigned a portion of the wheel. Each roulette wheel slice's size corresponds exactly to its fitness value. According to fitness value, an individual has a higher chance of being selected i.e., Individual with higher fitness value have more chances of selection. The wheel slice is calculated as

$$W_s = \frac{F^R}{\sum_{j=1}^{N} F_j^R} \tag{2}$$

where

$$F^R = (F_{\max} - F_k) + 1, \tag{3}$$

F^R is the reversed magnitude fitness function, F_j is the fitness of the jth chromosome, F_k is the fitness of the kth chromosome, $F_{\max}$ is the maximum fitness out of all the population's chromosomes, and N is the number of chromosomes in the population.

Crossover In this work, a one-point crossover operator is used to create offspring. In this crossover operator, one cut point is selected randomly at the same position from both the parent chromosomes and the right (left) section of the points is exchanged resulting in the formation of two offspring.

Mutation Exchange mutation (Banzhaf 1990) is used in the work to avoid premature convergence by giving new paths. The exchange mutation operator randomly selects two cities in the tour and exchanges them.

Termination condition The termination condition (or stopping criteria) is to stop the process of the genetic algorithms. The algorithm finds better solutions after a few iterations but this tends to stop working in the later phases when the changes aren't as large. So, we need to stop the process to make sure that the solution is close to the best one. In most cases, we preserve one of the following criteria for termination:

(a) When there hasn't been any change in the population for a while.
(b) When we have reached a certain, predetermined number of generations.
(c) When the objective function has attained a certain value that has already been set.

The number of iterations is used as a termination condition in this study.

6.2 Time and space complexity

The time complexity of the SGA algorithm as per the steps mentioned will be as under:

Initialization The initialization step involves setting up the parameters such as population size, crossover probability and mutation probability. So, it has a constant time complexity O(1).

Generate an Initial Population Generating the initial population involves creating a fixed number of candidate solutions, which typically have a linear time complexity O(n).

Calculate the Objective Function Evaluating the objective function for each candidate solution in the population requires calling the function n times. Thus, the time complexity for this step is O(n).

Selection Process Fitness proportionate selection or roulette wheel selection can be implemented efficiently with a linear time complexity of O(n). The process involves calculating the cumulative fitness and selecting individuals accordingly.

Crossover Operator The time complexity of a one-point crossover operator depends on the size of the candidate solution representation and hence the time complexity can be approximated as O(n).

Mutation Operator The exchange mutation operation also depends on the size of the candidate solution representation. So, the time complexity can be approximated as O(n).

Update Best Candidate Solution Updating the best candidate solution involves comparing the fitness of newly generated individuals to determine the best. This step has a linear time complexity O(n).

Output Best Candidate Solution The output step has a constant time complexity O(1) as it involves returning the best candidate solution found.

The overall time complexity of the SGA can be approximated as the sum of the complexities of each step. So, the SGA's overall expected time complexity is is $O(1)+O(n)+O(n)+O(n)+O(n)+O(n)+O(n)+O(1)= 2\times O(1)+6\times O(n) \approx O(n)$.

Similarly, the overall time complexity of the AGA, BGA and RCGA can be approximated as O(n^2).

Table 2 Statistical results of compared algorithms for formulated problem

	SGA				AGA				BGA				RCGA			
	Mean	S.D.	Best Value	ET	Mean	S.D.	Best Value	ET	Mean	S.D.	Best Value	ET	Mean	S.D.	Best Value	ET
T = 100	2044.15	126.67	1814	3.52	2009.35	195.1316	1767	2.75	1984.9	118.24	1742	13.75	1914.95	184.98	1448	17.13
T = 200	1792.05	105.52	1621	6.45	1822.3	145.86	1501	5.96	1692.3	105.06	1527	25.35	1669.6	137.69	1411	30.12
T = 300	1653.7	145.54	1407	9.47	1685.65	125.02	1438	8.50	1578.95	95.74	1389	29.41	1592.3	118.19	1368	37.14
T = 500	1539.15	142.20	1264	15.72	1591.2	94.52	1420	19.61	1517.5	95.72	1320	35.41	1521.8	114.78	1322	45.20
P = 100	2139.2	159.11	1881	4.43	2151.6	129.36	1944	3.29	2002.05	139.92	1721	16.13	2076.55	172.29	1733	17.74
P = 150	2143.45	156.45	1818	5.53	2180.35	172.33	1819	4.61	2019.1	138.07	1768	22.22	2143.45	171.78	1813	23.10
P = 200	2150.8	158.25	1720	6.36	2199.15	167.74	1961	6.03	2067.6	135.37	1856	28.48	2139.9	171.55	1754	32.17
P = 300	2235.9	161.18	1977	8.11	2150.4	126.11	1866	10.75	2111.7	148.89	1926	35.26	2243.15	149.88	1926	45.85
C = 25	1681.15	155.07	1380	21.85	1654	128.60	1345	2.29	2102.15	134.73	1848	13.04	1613.1	138.31	1376	22.17
C = 50	1606.85	124.97	1404	15.27	1597.95	115.93	1431	2.87	2076.4	142.43	1776	13.85	1491.5	132.55	1291	27.88
C = 75	1623.65	138.22	1347	29.41	1637.65	133.43	1407	3.51	1985.15	158.64	1701	14.62	1515.9	123.29	1312	31.12
C = 95	1617.75	129.63	1433	20.96	1563.9	106.07	1305	4.37	2016.25	142.61	1795	15.63	1544.2	95.55	1318	35.13

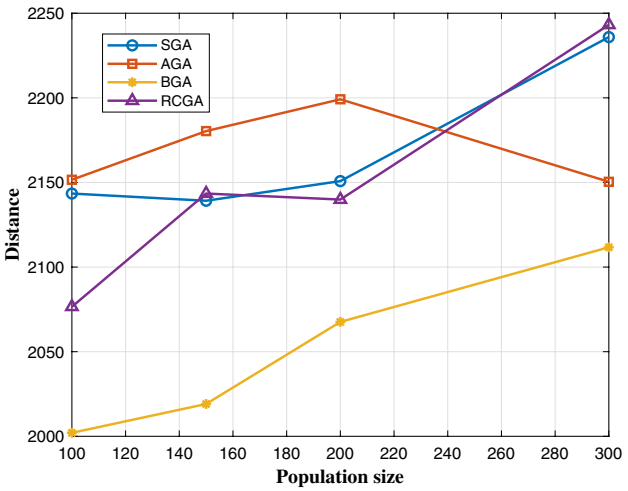

Fig. 1 Shortest distance of different algorithms at different population size

7 Results and discussion

In this section, we evaluate the performance of the SGA, AGA, BGA and RCGA approaches to obtain the optimal solution. The discussed algorithms were implemented in MATLAB R2021b, and their performances were tested using a laptop (computer) core i5-11300 H CPU @ 3.10GHz, with 16 GB of RAM. The performance of these four algorithms is analyzed by varying the parameters like population size, crossover and iteration. The results obtained by these algorithms on various parameters are listed in Table 2. Each algorithm is performed 20 times to obtain the minimum distance (Best value), the mean value, the standard deviation (S.D.) and the estimated time (ET).

In evolutionary computation, one of the most important factors to think about is an algorithm's population size (Mora-Melia et al. 2017) as it has a big effect on how well and quickly an algorithm works. So, it is foremost to investigate the performance of various genetic algorithms (SGA, AGA, BGA, RCGA) with fixed crossover and mutation at different population sizes. Figure 1 shows the performance of various GAs at different population sizes. The results are performed at fixed crossover, mutation, and iteration rates of 90, 5, and 100 respectively to evaluate the influence of the population size on these algorithms.

When the population size (p) increases, the minimum distance in each algorithm also increases. As we can see, in SGA at $p = 100$, $p = 150$, $p = 200$, and $p = 300$, the minimum distance is 2139.2, 2143.45, 2150.8, and 2235.9, respectively. Also, at $p = 100$, $p = 150$, $p = 200$, and $p = 300$ in the BGA, the minimum distance is 2002.05, 2019.1, 2067.6 and 2111.7 respectively. Similarly, in AGA and RCGA, as the population size increases, so does the minimum distance. So, the behavior of all of these algorithms

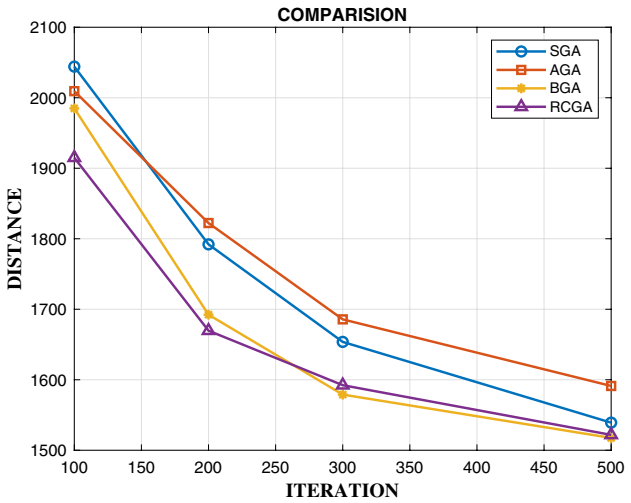

Fig. 2 Shortest distance of different algorithms at different iteration

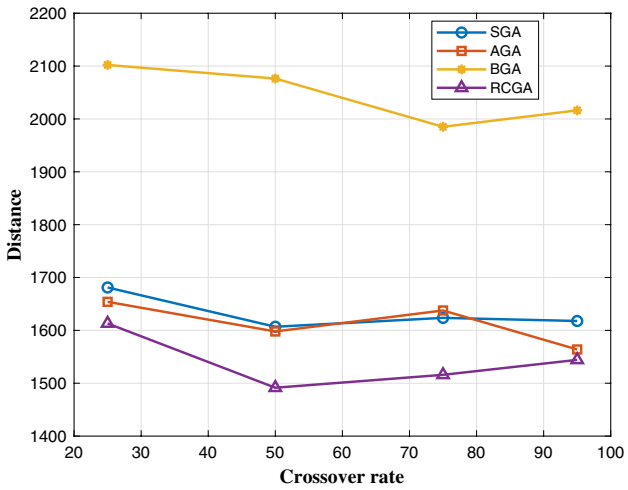

Fig. 3 Shortest distance of different algorithms at different crossover

appears to be similar as the minimum distance increases with population size. It means that to find the minimum distance, we have to consider the smallest population size. Therefore, the performance of these algorithms is analyzed at a population size of 100 to see which one provides a better optimal solution. At $p = 100$, the minimum distances of SGA, AGA, BGA, and RCGA are 2139.2, 2151.6, 2002.05, and 2076.55, respectively. In this case, BGA provides a better optimal solution (minimum distance) with a path $4 \rightarrow 17 \rightarrow 7 \rightarrow 9 \rightarrow 18 \rightarrow 1 \rightarrow 22 \rightarrow 20 \rightarrow 15 \rightarrow 16 \rightarrow 5 \rightarrow 12 \rightarrow 11 \rightarrow 14 \rightarrow 19 \rightarrow 2 \rightarrow 21 \rightarrow 10 \rightarrow 6 \rightarrow 8 \rightarrow 13 \rightarrow 3$..

Now we investigate the effect of iteration on these algorithms by varying the size in the range of 100–500. The higher iteration size increases the chance of getting a better solution as the solution space is searched more thoroughly. With each passing iteration, the bad quality solutions get discarded, and the better solutions participate in the search process. To investigate the performance of various genetic algorithms, the results are performed at fixed rate of crossover, mutation, and population size of 50, 10, and 100 respectively. Figure 2 shows that when the iteration increases, the minimum distance in each algorithm decreases. In SGA, at iteration = 100, 200, 300, and 500, the minimum distances are 2044.15, 1792.05, 1653.7, and 1539.15, respectively. In AGA, at iteration = 100, 200, 300, and 500, the minimum distances are 2009.35, 1822.3, 1685.65,and 1591.2 respectively. Similarly in other algorithms, as iteration increases, the minimum distance decreases. So, the behavior of all of these algorithms appears to be similar as the minimum distance decreases with increasing of iteration. It means that to find the minimum distance, we have to consider the higher iteration.

Therefore, the performance of these algorithms is analyzed at the iteration, 500 to see which one provides a better optimal solution. At iteration = 500, the minimum distance of SGA, AGA, BGA, and RCGA is 1539.15, 1591.2, 1517.5, and 1521.8 respectively. In this case, BGA provides a better optimal solution (minimum distance). So, in all of these algorithms, BGA also gives a better optimal solution in terms of iteration with a path $10 \rightarrow 14 \rightarrow 21 \rightarrow 20 \rightarrow 19 \rightarrow 1 \rightarrow 2 \rightarrow 22 \rightarrow 11 \rightarrow 12 \rightarrow 9 \rightarrow 18 \rightarrow 6 \rightarrow 5 \rightarrow 16 \rightarrow 15 \rightarrow 17 \rightarrow 7 \rightarrow 3 \rightarrow 4 \rightarrow 8 \rightarrow 13$.

Now we investigate the effect of crossover on various genetic algorithms analyzed by varying it. It is well known that the selection of crossover rate is crucial to the effectiveness of genetic algorithms. In the past, various research have been worked on determining optimal crossover or mutation rates which is still a problem as it depends on problems and even for different stages of the genetic process. So, here we applied various genetic algorithms on our problem to see the effect of the rate of crossover. The results are performed at fixed iteration (100), population size (100) and mutation rate (5) with crossover rates of 25, 50, 75, and 95, i.e., crossover probability 0.25, 0.50, 0.75, 0.95. The performance of these algorithms is analyzed using Fig. 3 to see which one provides a better optimal solution in terms of crossover rate.

At crossover = 25, the minimum distance for SGA, AGA, BGA, and RCGA, respectively, is 1681.15, 1654, 2102.15, and 1613.1. In this case, RCGA provides a better optimal solution (minimum distance). Similarly, at the rate of crossover = 50, the minimum distances of SGA, AGA, BGA, and RCGA are 1606.85, 1597.95, 2076.4 and 1491.5 respectively. In this case, RCGA also provides a better optimal solution. As a result, RCGA provides a better optimal solution in all these algorithms in terms of crossover with a path $15 \rightarrow 4 \rightarrow 3 \rightarrow 13 \rightarrow 16 \rightarrow 21 \rightarrow 11 \rightarrow 19 \rightarrow 8 \rightarrow 22 \rightarrow 9 \rightarrow 2 \rightarrow 17 \rightarrow 7 \rightarrow 1 \rightarrow 5 \rightarrow 12 \rightarrow 18 \rightarrow 10 \rightarrow 14 \rightarrow 6 \rightarrow 20$.

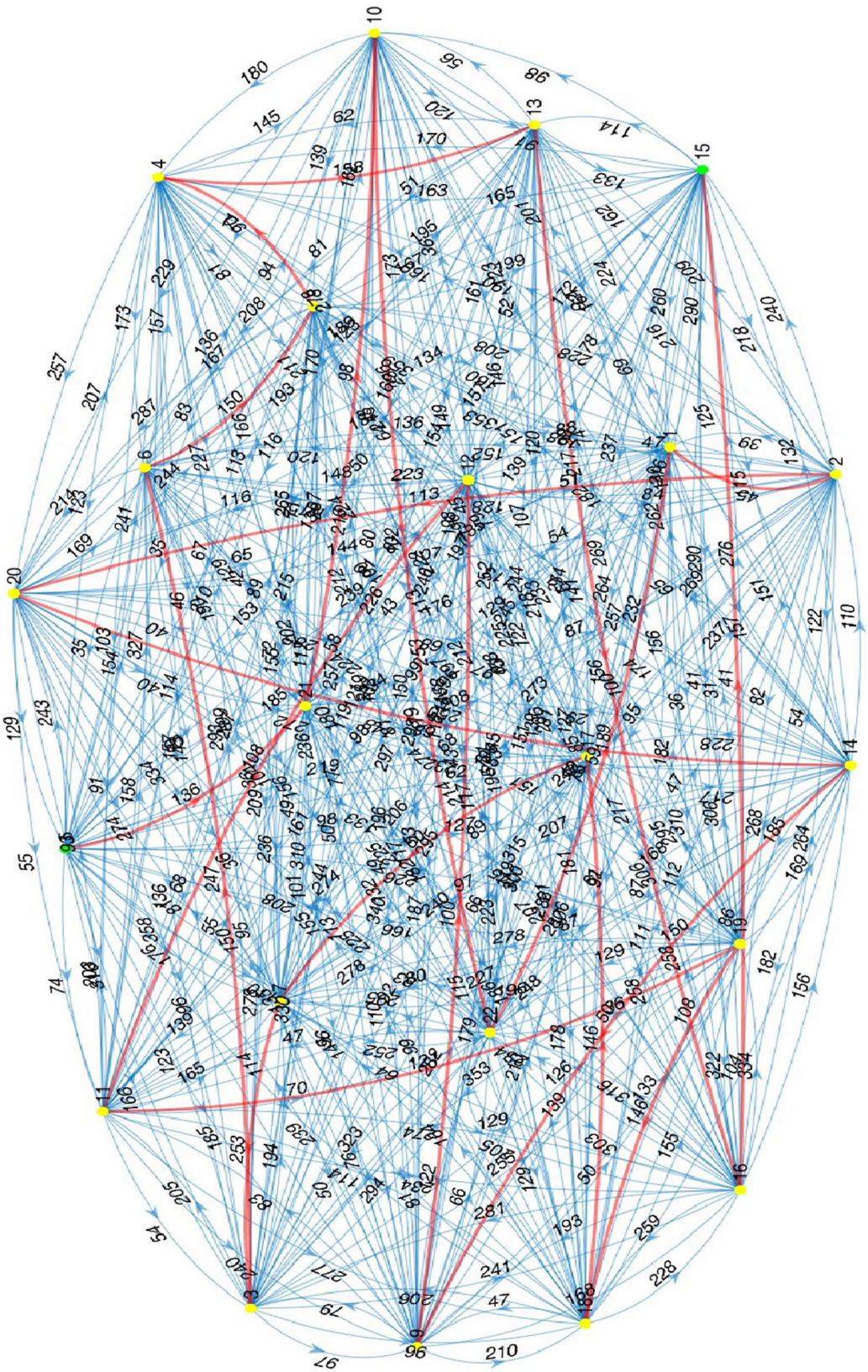

Fig. 4 Highlighted the shortest path in a formulated ATSP problem

The results discussed above are based on their mean and standard deviation values. However, the best minimum distance is 1264 provided by the SGA at iteration 500 and the corresponding path has been highlighted in Fig. 4.

8 Conclusion

In the present work, we have discussed genetic algorithm and its variants on a asymmetric TSP as our aim was to compare the quality of solutions at different parameters. Based on the results of various GAs on the given ATSP, the real-coded genetic algorithm is superior in terms of the rate of crossover. The binary genetic algorithm performs better in finding the optimal solution in terms of iteration and population size. The major and minor variations in the distance are observed in the parameters such as size of population and the number of iterations. This study presents a comparative analysis of various genetic algorithms on ATSP, which is realistic and relevant for GA's adaptability in real-world applications. Also, this study suggests various research directions for future research, one of which involves developing hybrid or improvement versions of GA to apply the ATSP problem.

Acknowledgements The first author is extremely grateful to the Council of Scientific & Industrial Research (CSIR) for providing Junior Research Fellowship (JRF) with file number 09/1152(0024)/2020-EMR-1 and encouragement, which made this research possible.

Declarations

Conflict of interest There are no conflicts of interest to disclose by the authors in relation to the current study.

Ethical approval The author did not conduct any studies involving human participants or animals for this article.

Informed consent All the authors have approved the manuscript and agree with its submission to the International Journal of System Assurance Engineering and Management.

References

Adewumi AO, Adeleke OJ (2018) A survey of recent advances in vehicle routing problems. Int J Syst Assur Eng Manage 9:155–172

Akhand MAH, Ayon SI, Shahriyar SA, Siddique N, Adeli H (2020) Discrete spider monkey optimization for travelling salesman problem. Appl Soft Comput 86:105887

Ali MZ, Awad NH, Suganthan PN, Shatnawi AM, Reynolds RG (2018) An improved class of real-coded genetic algorithms for numerical optimization. Neurocomputing 275(1):155–166

Ascheuer N, Junger M, Reinelt G (2000) A branch & cut algorithm for the asymmetric traveling salesman problem with precedence constraints. Comput Optim Appl 17:61–84

Balas E, Christofides N (1981) A restricted Lagrangean approach to the traveling salesman problem. Math Program 21(1):19–46

Bansal N, Singh AK (2022) Valuable survey on scheduling algorithms in the cloud with various publications. Int J Syst Assur Eng Manag 13(5):2132–2150

Bellmore M, Nemhauser GL (1968) The traveling salesman problem: a survey. Oper Res 16(3):538–558

Boryczka U, Szwarc K (2019) The harmony search algorithm with additional improvement of harmony memory for asymmetric traveling salesman problem. Expert Syst Appl 122:43–53

Buriol L, Franca PM, Moscato P (2004) A new memetic algorithm for the asymmetric traveling salesman problem. J Heurist 10:483–506

Carpaneto G, Dell'Amico M, Toth P (1995) Exact solution of large-scale, asymmetric traveling salesman problems. ACM Transact Math Softw (TOMS) 21(4):394–409

Deep K, Mebrahtu H, Nagar AK (2018) Novel GA for metropolitan stations of Indian railways when modeled as a TSP. Int J Syst Assur Eng Manage 9:639–645

Deepa SN (2008) Introduction to genetic algorithms. Springer, Berlin

Dehedkar SN, Raj S (2022) Determination of optimal location and implementation of solar photovoltaic system using ETAP. In: 2022 IEEE 2nd International Symposium on Sustainable Energy, Signal Processing and Cyber Security (iSSSC). IEEE, pp 1–4

Eshelman LJ, Schaffer JD (1993) Real-coded genetic algorithms and interval-schemata. Found Genet Algorithm 2(1):187–202

Fiechter CN (1994) A parallel tabu search algorithm for large traveling salesman problems. Discret Appl Math 51(3):243–267

Gary MR, Johnson DS (1979) Computers and intractability: a guide to the theory of NP-completeness

Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison Wesley, Reading

Goldberg DE (1991) Real-coded genetic algorithms, virtual alphabeths, and blocking. Complex Syst 5(1):139–167

https://www.google.com/maps/@29.282929,76.026532,8z

Katoch S, Chauhan SS, Kumar V (2021) A review on genetic algorithm: past, present, and future. Multimed Tools Appl 80(5):8091–8126

Kim JW, Kim SW, Park P, Park TJ (2002) On the similarities between binary-coded GA and real-coded GA in wide search space. Proc 2002 Congress Evoluti Comput 1(2):681–686

Larranaga P, Kuijpers C, Murga R (1999) Genetic algorithms for the travelling salesman problem: a review of representations and operators. Artif Intell Rev 13(2):129–170

Li K, Zhuo Y, Luo X (2022) Optimization method of fuel saving and cost reduction of tugboat main engine based on genetic algorithm. Int J Syst Assur Eng Manage 13(1):605–614

Li W, Wang C, Huang Y, Cheung YM (2023) Heuristic smoothing ant colony optimization with differential information for the traveling salesman problem. Appl Soft Comput 133:109943

Lin C (2009) An adaptive genetic algorithm based on population diversity strategy. In: 2009 Third International Conference on Genetic and Evolutionary Computing, pp 93–96

Mahapatra S, Dey B, Raj S (2021) A novel ameliorated Harris hawk optimizer for solving complex engineering optimization problems. Int J Intell Syst 36(12):7641–7681

Mahapatra S, Raj S (2023) A novel meta-heuristic approach for optimal RPP using series compensated FACTS controller. Intell Syst Appl 18:200220

Majumdar J, Bhunia AK (2011) Genetic algorithm for asymmetric traveling salesman problem with imprecise travel times. J Comput Appl Math 235(9):3063–3078

Mohebifar A (2006) New binary representation in genetic algorithms for solving TSP by mapping permutations to a list of ordered numbers. WSEAS Transact Comput Res 1(2):114–118

Mora-Melia D, Martinez-Solano FJ, Iglesias-Rey PL, Gutierrez-Bahamondes JH (2017) Population size influence on the efficiency of evolutionary algorithms to design water networks. Procedia Eng 100(186):341–348

Mzili T, Mzili I, Riffi ME (2023) Artificial rat optimization with decision-making: a bio-inspired metaheuristic algorithm for solving the traveling salesman problem. Decis Making Appl Manage Eng

Nagata Y, Soler D (2012) A new genetic algorithm for the asymmetric traveling salesman problem. Expert Syst Appl 39(10):8947–8953

Osaba E, Del Ser J, Sadollah A, Bilbao MN, Camacho D (2018) A discrete water cycle algorithm for solving the symmetric and asymmetric traveling salesman problem. Appl Soft Comput 71:277–290

Osaba E, Yang XS, Diaz F, Lopez-Garcia P, Carballedo R (2016) An improved discrete bat algorithm for symmetric and asymmetric traveling salesman problems. Eng Appl Artif Intell 48:59–71

Pekny JF, Miller DL (1990) A parallel branch and bound algorithm for solving large asymmetric traveling salesman problems. In: Proceedings of the 1990 ACM annual conference on Cooperation, pp 56–62

Potvin JY (1996) Genetic algorithms for the traveling salesman problem. Ann Oper Res 63(3):337–370

Raj S, Bhattacharyya B (2018) Optimal placement of TCSC and SVC for reactive power planning using Whale optimization algorithm. Swarm Evol Comput 40:131–143

Raj S, Mahapatra S, Babu R, Verma S (2023) Hybrid intelligence strategy for techno-economic reactive power dispatch approach to ensure system security. Chaos, Solitons Fractals 170:113363

Rocha Y, Subramanian A (2023) Hybrid genetic search for the traveling salesman problem with hybrid electric vehicle and time windows. Comput Operat Res 155:106223

Saptarini NGAPH, Ciptayani PI, Wisswani NW, Suasnawa IW (2020) Adaptive genetic algorithm for high school time-table. J Phys 1569(3):01–06

Singh G, Gupta N, Khosravy M (2015) New crossover operators for real coded genetic algorithm (RCGA). In: 2015 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), pp 135–140

Tawhid MA, Savsani P (2019) Discrete sine-cosine algorithm (DSCA) with local search for solving traveling salesman problem. Arab J Sci Eng 44(4):3669–3679

Wang J, Huang J, Rao S, Xue S, Yin J (2008) An adaptive genetic algorithm for solving traveling salesman problem. In: International Conference on Intelligent Computing, pp 182–189

Wang J, Zhang M, Ersoy OK, Sun K, Bi Y (2019) An improved real-coded genetic algorithm using the Heuristical normal distribution and direction-based crossover. Comput Intell Neurosci 2019(1):01–18

Zbigniew M (1996) Genetic Algorithms+ Data Structures= Evolution Programs. Springer-Verlag, Berlin

Zhang T, Zhou Y, Zhou G, Deng W, Luo Q (2023) Discrete Mayfly algorithm for spherical asymmetric traveling salesman problem. Expert Syst Appl 221:119765