Contrasting Evolutionary Algorithm and Social Spider Algorithm using Travelling Salesman Problem

Eman Arif[1*], Asma Sanam Larik[1], Tasneem Adnan[2], Maryam Raees Ahmed[1]

[1]Faculty of Information and Computing Sciences, Dawood University of Engineering and Technology, Karachi-74800, Pakistan
emanarif656@gmail.com,asma.sanam@duet.edu.pk, m.raeesahmed3@gmail.com

[2]School of Mathematics and Computer Science, Institute of Business Administration, Karachi, Pakistan
E-mail: t.adnan.27499@khi.iba.edu.pk

Corresponding Author: emanarif656@gmail.com

Abstract: Combinatorial optimization problems have plenty of attention, both for the reasons of their complexity and for their practical applications. Looking at the range of metaheuristic techniques, there are two which have shown good results: the Genetic Algorithms (GA) and swarm-based techniques. A comparative study of the Genetic Algorithm and the Social Spider Algorithm (SSA) is described in this paper with Travelling Salesman Problem (TSP) as benchmark. The SSA, which is based on cooperative interaction mechanisms among the agents, is modified to address the TSP and its performance is compared to GA. The algorithms are contrasted with respect to the quality of solutions and efficiency of computation. The experimental results demonstrate that the convergence property and runtime efficiency of Social Spiders improves the problem considered. The paper is arranged as follows: In Section 1 and 2, is about introduction to the paperwork discussion of literature gaps section 3 we introduce the concept of Genetic algorithm and Swarm While Section 4, the problem is formulated. Section 5 focuses on implementation of Social Spider Algorithm and 6 presents experimental results and comparison of the two algorithms at hand: Genetic, Social Spider Optimization. Finally, in Section 6 conclusions are drawn up and prospects are discussed.

Keywords: Social Spider, Optimization, Evolutionary Algorithm, Swarm Intelligence.

I. INTRODUCTION

Combinatorial optimization problems are significant in the field of computer science and in the real world like routing, scheduling and network design. The problems among them are NP-hard and one that is widely studied is the Travelling Salesman Problem (TSP). In more cities, the calculation time of the optimal solution isn't feasible anymore, and the optimal solutions are therefore often solved using the metaheuristic algorithms, but not exactly.

How to use Evolutionary Methods (GA) to solve complex optimization problems, which are based on natural selection and have been widely applied. GA is evolving the candidate solutions iteratively using operators such as selection, crossover and mutation, and has proven to be good global searcher. Due to their effectiveness, Genetic Algorithms are often used as benchmark approaches in comparative optimization studies.

Many other powerful metaheuristics, based on the finding that agents exhibit collective behavior and interacts, can be classified under the swarm intelligence-based algorithms. Social Spider Algorithm (SSA) is a swarm-based optimization algorithm, which uses the idea of cooperative interactions among individuals to steer the search process to the promising areas of the solution space. SSA has demonstrated competitiveness in solving several optimization problems.

Even though both optimization and SSA have been successfully applied to certain problems, to our knowledge, not much comparative analysis has been done that gives general insight into their performance when applied to classical combinatorial problems such as the TSP. So, this research conducts comparative study of Genetic Algorithm and Social Spider Algorithm, by taking the Travelling Salesman Problem as a reference. The quality of the solutions delivered by the algorithms is compared, and their computational efficiency level is compared to determining the comparative advantages and their type of performance.

II. LITERATURE REVIEW

It has been called Metaheuristic Optimization (MHO) [1] the combination of the most complicated combinatorial problems (molecular dynamics simulations etc.) and the use of the general-purpose black box physical simulations of last few years. The first-generation evolutionary computing approaches were responsible for the concept of population based approaches, crossover and mutation approaches, and probabilistic selection approaches for the evolving algorithms used today, and evolved the concept of global search and robust across a variety of problem classes [2],[3]. While flexible, conventional GA and evolutionary strategies

have already been used to some TSP problems successfully, they too can converge slowly for large scale TSP problems and are not computationally efficient [4] [5]; thus, other paradigms of optimization exist.

Collective behavior of social insects and animals has been an essential ground for decentralized optimization based on biologically inspired models known as swarm intelligence. Some of the algorithms which take advantage of emergent behavior, distributed cooperation and self-organization in the way of navigating in the complex search spaces are called Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Bacterial Foraging Optimization, Firefly Algorithm, Grey Wolf Optimizer and Slap Swarm Optimization [6, 1, 7, 8, 9]. Comparative studies and reviews reveal a trend towards hybridization, adaptive parameter control, parallelization to provide better scalability and solution quality for the large size of problem instances [10, 11, 27]. Compared to classical evolutionary methods, these methods can respond to the trade-off between exploration and exploitation in a more efficient way, thus becoming more interesting for NP-hard combinatorial problems such as the TSP [4, 5].

One of the recently developed swarm-based algorithms inspired by the real-life collaborative foraging strategy of social spider colonies that uses vibrations for communication is the Social Spider Algorithm (SSA) [12], [13], [14]. Trials in SSA share memories via web vibrations (positions, fitness data) that guide adaptive motions towards specific regions of interest in the solution space. The spiders' behaviors, mating structure, and weight schemes help to preserve the population's diversity and prevent premature convergence [12], [15], [14], [16] with males and females being distinctively modeled. Large-scale TSPs can be more efficiently solved using some variants of SSA such as population-mining systems [17, 18, 19, 20, 16] than conventional evolutionary algorithms and other swarm-based systems.

When used with the TSP [17] [12] the solution quality, convergence speed and robustness are unique. Comparative studies [16] have always been able to demonstrate SSA's convergence is faster than GA's convergence and the length of the generated tours are shorter, especially in the case of the Qatar TSP dataset and Concorde solver benchmarks [21], [22]. Hybrid variations of SSA models that contain components of local search heuristics like 2-OPT, 3-OPT or more genetic algorithms (GA), are somewhat more sophisticated and have led to further solution stability and convergence [17, 23, 24]. The results from benchmarking SSA with other metaheuristics, such as Spider Monkey Optimization, Lion Optimization, Grey Wolf Optimizer, Artificial Algae Algorithms, Earthworm Optimization, and Memetic hybrids in various problem scales further support its competitive performance and adaptability [23, 10, 24, 25, 9].

Even with the advances of knowledge about metaheuristics, the study is currently underway to combine them with machine learning, and recently, near-optimal TSP tours for the TSP problem were predicted using machine learning with Graph Convolutional Networks (GCNs) that can be used as high-quality initializations for the swarm-based optimizers [26]. Such learning-aided metaheuristics result in tremendous improvements in terms of scalability and efficiency, especially in high dimensional versions of the TSP, representing a broader evolution of learning-hybrid approaches that combine data-driven models with nature-inspired approaches [11, 26]. Additionally, large-scale TSP instances are solved within reasonable time using a parallelization and/or distributed computing method without blurting the quality of the solution [19, 27].

Moreover, empirical evidence reveals that adaptive SSA variants have strengths for finding local (suboptimal) minima and preserving diversity, particularly in problems with extremely deceptive fitness functions and/or edge cost distributions [18, 20, 16]. Compared to ACO, PSO (10) and SSA [12, 13, 6] have been able to explore the best regions faster with maintaining the capability of exploring without sacrificing any of this process when compared to the ACO, despite the fact that they explored at the same speed with ACO, allowing a good balance and showing better performance when applied to the combinatorial optimization problems. Cooperative partitioning of the structure of the population, the use of self-adaptive control parameters and hybrid local search heuristics further improve the performance for large TSP problems, with the number of nodes exceeding thousand, for recently developed versions of parallel and hybrid SSA algorithms [19, 16, 27].

Overall, the literature shows that various variations of SSA are a potentially strong and useful procedure for solving TSP. With its design component biologically inspired communication method, adaptive weight schemes and in combination with learning models [12, 17, 18, 19, 26, 14, 16]—SSA always delivers high-quality solutions no matter how small, medium or large the instances are. With its flexibility and competitiveness in comparison to other metaheuristic methods, and newly developed parallelization and machine learning methods, SSA is a novel and current algorithm to solve combinatorial optimization problems such as TSP [12], [21], [23], [26], [27].

III. BACKGROUND

Swarm Intelligence

Bonabeau defines swarm intelligence as the "any attempt to design algorithms or distributed problem-solving devices inspired by the collective behavior of the social insect colonies and other animal societies" [6]. Self-organization and division of labor are the principal components of swarm intelligence. These two components are imperative in carrying out intelligent behavior. Everyone is only disturbed by a local stimulus and is able to perform their own global task, at the same time the division of labor makes everyone his/her own supervisor, preventing the overall centralized supervision [13].

The cooperative entities, in swarm intelligence, are composed of individuals, with everyone carrying out cooperative tasks, and reproducing specialized behaviors based on their biological specifications. However, the segregated division of labor between the sexes is ignored in some swarms that function regardless of this distinction or in swarms which are asexual, for example, eliminating the opportunities of 'adding new and selective operators' [13]. Social insects or animals with interesting behaviors have been attracting the interest of the scientists who try copying the foraging behavior of such a group to perform real world computations and solve optimization problematics. Besides offering survival benefits, the organizational nature of social insects has also been effective in performing complicated functions even with a few limited pieces of information and behavioral rules offered. [13]

Genetic Algorithm (GA)

For a long time now, the genetic algorithm has been applied to the solving of optimization problems. These are referred to as function optimizers. This algorithm chooses a chromosome like structure in order to store a possible solution to the specific problem and imposes critical information on the chromosome via recombination operators as being crucial to the solution [2].

The algorithm begins with a random start (random initial chromosome), which is evaluated and reproduced so that the chance of better solutions to the problem desired is increased over the course of the algorithm whilst the chance of worse solutions to the problem desired is decreased.

1. PROBLEM FORMULATION

Social Spider Algorithm (SSA) can be applied in solving optimization problems. The problem being worked on is the phenomenal 'Travelling Salesman Problem' and the performance of SSA will be evaluated on this problem assuming this has wide range of applications. For instance, travel tours can use this optimization technique to plan the routes of the tour, in order to cover most of the cities in minimal time period.

Dataset

The Qatar Dataset [21] would be used for testing the Travelling Salesman Problem as the benchmark for the evaluation of the Social-Spider Algorithm. The data set consists of 194 cities, with all cities presumed to be linked to each other, constituting a complete city network. The algorithm is used to find the shortest possible route that involves travelling through all the cities in the data set. One such route is indicated in Fig. 1 [21].

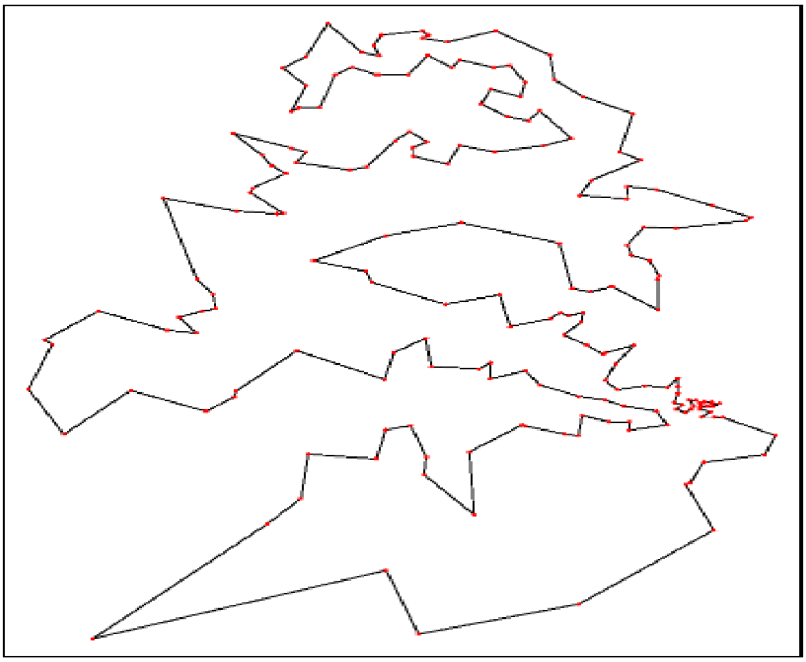

Fig. 1. One of the routes in the Qatar Dataset. Adapted from [7].

Time Constraint

Although evolutionary algorithm can be applied to solve the Travelling Salesman Problem, it is still necessary to minimize the time constraint. If there is more data, the genetic algorithm is very time consuming and not very efficient. Compared with the SSA, which gives an efficient computational time, however.

2. SOCIAL SPIDER OPTIMIZATION FOR TRAVELLING SALESMAN PROBLEM

The Social Spider Algorithm is a relatively recent algorithm based on the foraging behavior of social spiders [12] which is a metaheuristic. This motion towards the place where prey is set among the web is referred to as collective and collaborative foraging. The researchers found that the spiders communicate by vibrating and receive and interpret such vibrations to locate the food and its direction [15]. The optimization in Social Spider Algorithm over the search space using this collaborative behavior is inspired from the cooperative nature of one spider with other spiders to move towards the prey [12].

The Social Spider Optimization approach proposes that the search space be considered as the total web of a spider colony with spiders placed at random locations across the web. A position is a possible solution of the target problem which each social spider

has on the web. In this case, the position represents a possible tour of the cities which can be the solution to the Travelling Salesman Problem.

Male and Female Spiders

Social spiders have two castes, males and females. This is because 70 to 90 percent of the total spider population made up of female spiders and 10 percent composed of males.

The number of female spiders randomly selected from the population is selected according to formula:

$$N_f = \text{floor } [(0.9 - \text{rand } 0.25N] \quad (1)$$

While the number of males is calculated as:

$$N_m = N - N_f \quad (2)$$

- Where N = Size of the total population. Each spider in the population is an agent which brings about metaheuristic optimization. A population of worms keeps the following information about each spider:
- Boolean: A type of value that can be either true or false.
- Position: The position of the spider indicates that the spider has completed an entire tour - this is one possible candidate solution.
- Weight: The weight of the spider basically refers to the fitness of spider where the length of the tour (position n) corresponds to the weight attribute.
- The radius equals the distance from which a spider can communicate with other spiders.

Fitness

Each of the spider has a fitness, that is, the weight of each, which is evaluated based on the length of his tour and compared with the shortest and the longest of the search space of all spiders, without gender distinctions, respectively. The weight of each spider is given by the formula above.

$$w_i = \frac{J(s_i) - worst_s}{best_s - worst_s} \quad (3)$$

Where, $J(s_i)$ represents the length of the tour of spider S_i while the best and the worst represent the longest and the shortest tour respectively, which are calculated as:

$$best_s = \max_{kc\{1,2,\ldots,N\}} (J(s_k)) \text{ and } worst_s = \min_{kc\{1,2,..,N\}} (J(s_k)) \quad (4)$$

The minimization aim is achieved by making the fitness of each spider reciprocal, i.e., the worst (smallest tour position) spider gets the largest weight while the best (longest distance) spider obtains the smallest weight.

Vibrations

A web systematic communicating mechanism is utilized by the spiders. Their means of sharing information is through the generation and reception of vibrations. The Social Spider Algorithm uses three different kinds of vibrations in order to communicate information to the other members in the web as shown in fig 2 [13]. These vibrations are directly proportional to the distance and weight of the spider: a spider is in the area near someone will experience stronger vibrations than another spider located at a distance away. The SSA takes advantage of the various vibrations as intended by the following:

1. **Vibration C:** this vibration transmits information to all the members *i* such that the communicating spider is the nearest spider to *i* and possess a higher weight than *i*. It is computed as:

$$Vibc_i = w_c . e^{-d_{i,c}^2} \quad (5)$$

Where w_c is the weight of the nearest spider and di,j is the distance between the two vectors of the spider positions (that is their tour distance's difference).

2. Vibration F: this vibration transmits information to the members i such that the communicating spider is the nearest female spider to i. It is computed as:

$$\boldsymbol{Vibf_i = w_f . e^{-d_{i,f}^2}} \quad (6)$$

Where w_f is the weight of the nearest female spider

3. Vibration B: this vibration transmits information from b to all the members i such that the communicating spider b has the highest weight among all the spiders in the population. It is computed as:

$$\boldsymbol{Vibb_i = w_b . e^{-d_{i,b}^2}} \quad (7)$$

Where w_b is the weight of the spider with best fitness in the population.

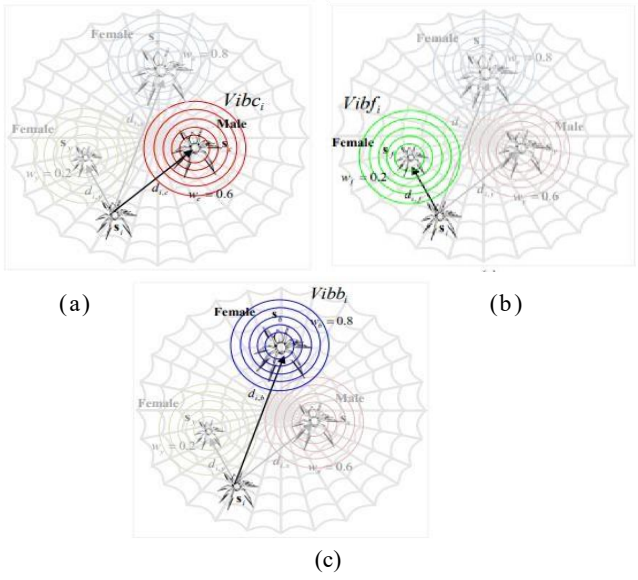

Fig 2: (a) [left] Vibration C **(b)** [right] Vibration F **(c)**[center] Vibration B Adopted form [3]

Initial Population

The initial population of spiders is generated using the calculated number of male and female spiders. Each spider is assigned its gender and a randomly initialized position representing a candidate tour for the TSP.

Cooperative Spiders

Spiders also exhibit cooperative behavior, the male and female members of the population. Cooperative operator in female, does calculation of vibration C, and does calculation of vibration B with population members (males and females). If the random number r is greater than the threshold, then depending on the vibrations experienced by the spider, and the value of r, the new position of that spider, that is, the tour, is generated which is influenced by that particular spider in the population.

Conversely, males perform cooperative action only if the female nearest to them emits more Vibration However, in the case of the male population the cooperative behavior is performed only in response to female that emits Vibration F the most near. The dominant male members will then shuffle their tours (as random number r is greater than the randomly chosen threshold) due to influence from the female spider, introducing mutation in the original solution, and thus variation in the candidate solution.

Mating Operator

The mating operation in the social spider optimization algorithm is carried out by the dominant males and females of the population. All the males whose weights are greater than the median weight of the males in the population make up the dominant males.

Every dominant male then searches within his radius for females and copulates with them, thus replacing any weak males he encounters within his radius, if the weight of the new males is more than the weight of the weakest other male. In this case, the formula for calculating radius is the same as shown above:

$$\boldsymbol{r = \frac{\sum_{j-1}^{n}(p_j^{high} - p_j^{low})}{2 .n}} \quad (8)$$

cities and are the max index and min index of the cities to be visited in the tour, n is the number of cities.

During the entire process of mating the weight of the individual is very important in determining how much influence it will have in the offspring (new brood of the spider). Using the weight of the spider, the probability of the influence that it will have on the new brood is calculated as:

$$P_{s_i} = \frac{w_i}{\sum_{jcT^2} w_f}$$

Where represents the weight of a particular spider while the denominator consists of the sum of weights of all the spiders in the population. The position of the offspring depends on the probabilities (using a roulette wheel method).

The mating process of Social Spider Algorithm is represented in following Table 1. Following Table 1 is used to represent the mating process under S. Algorithm. The position indicated in the Table, however, is merely a representation of the position and does not represent the actual position of the Travelling Salesman Problem.

Table 1: Data for constructing the new spider S_{NEW} through the roulette method

Spider	Position	w_i	p_{si}
s_1 (f_1)	(-1.9, 0.3)	0.00	—
s_2 (f_2)	(1.4, 1.1)	0.57	0.22
s_3 (f_3)	(1.5, 0.2)	0.42	0.16
s_4 (f_4)	(0.4, 1.0)	1.00	0.39
s_5 (f_5)	(-1.0,-1.5)	0.78	—
s_6 (m_1)	(-1.3,-1.9)	0.28	—
s_7 (m_2)	(0.9, 0.7)	0.57	0.22
s_8 (m_3)	(0.8,-2.6)	1.42	—
S^R_{EW}	(0.9, 1.1)	1.00	—

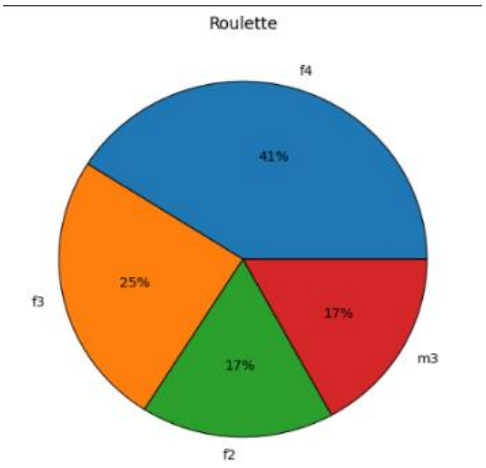

Fig. 3. Visualization of data for constructing the new spider SNEW through the roulette method.

IV. RESULTS AND DISCUSSION

In this section, we introduce the results of the Social Spider Algorithm, as compared with the benchmark Evolutionary Algorithm, namely the Genetic Algorithm. The performance assessment of the two algorithms compared to each other, as well as a collective analysis of the simulation results, is carried out.

When this Social Spider Algorithm method is modelled and implemented for solving Travelling Salesman Problem, it is observed that the computational results are yielding very similar answer for comparatively small dataset; and the time duration is really

negligible while executing it. On the other hand, once a larger data set like Qatar data set (more than 190 cities) is used for testing the 2 algorithms regarding their computational efficiency, the results obtained by the Genetic algorithm is much slow as compared to output of Social Spider Algorithm. The Social Spider Algorithm results in the shortest path as soon as they are found; after each iteration it provides the length of the shorted path in less than 20 seconds. Whereas the genetic algorithm, when ran on the same input data set computes, the best fit route at each iteration in more than 75 seconds.

It was noticed however, that after a few generations of the Social Spider Algorithm, it will produce the same results for a few generations in a row before generating the next best fit route along with making some premature convergence. The convergence rate of the Social Spider Algorithm is very rapid, however.

The experimental results of the two algorithms, when each ran for 500 generations, same population size and data set are summarized in the table as follows

Table 2: Comparing the experimental results of the Social Spider and Genetic Algorithm

Generation	Social Spider	SSA Runtime	Genetic Algorithm	GA Runtime
0	85313	0:01:08	83978	0:01:08
50	84754	0:04:51	81027	0:05:37
100	80943	0:07:21	80551	0:09:37
150	75744	0:09:48	79967	0:11:52
200	75604	0:12:56	76762	0:15:24
250	74357	0:17:40	73742	0:17:31
300	72787	0:21:23	76418	0:20:03
350	70783	0:20:05	75862	0:22:23
400	70454	0:22:46	74649	0:27:13
450	68820	0:26:36	74429	0:27:23
500	66502	0:29:46	71788	0:30:11
550	61171	0:32:23	69000	0:33:45
600	57908	0:35:15	69638	0:36:49
650	51751	0:37:31	66602	0:38:56
700	47683	0:42:13	65058	0:43:46
750	46658	0:42:25	65058	0:43:46
800	45389	0:47:40	65058	0:49:00
850	39262	0:49:03	65058	0:49:52
900	32058	0:52:26	65058	0:55:46
950	32058	0:55:31	65058	0:59:54
1000	32058	0:55:31	65058	0:59:54

Twenty-two below is a computational convergence graph of the two algorithms. Even interesting is that the genetic algorithm delivers the same results as the Social Spider Optimization Algorithm although it is slower in delivering the results than the Social Spider Algorithm.

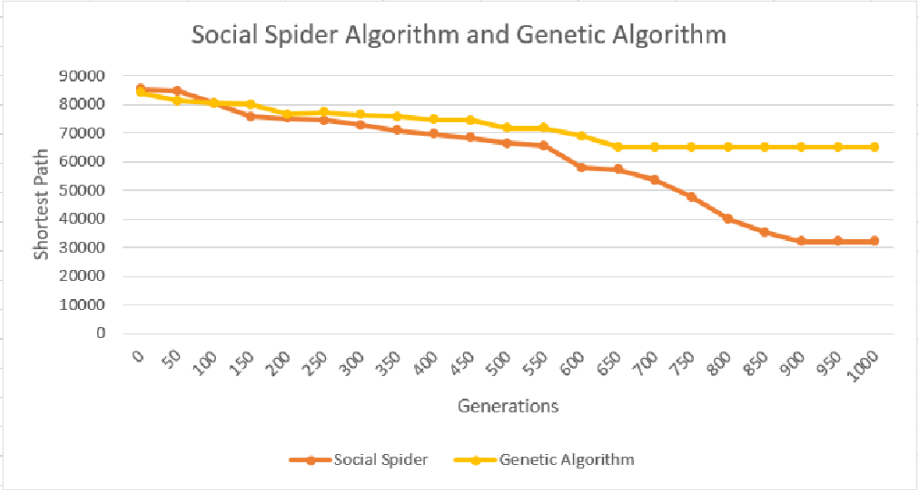

Fig 3: Graph analyzing the Computational convergence of the Genetic Algorithm and the Social Spider Optimization Algorithm

V. CONCLUSION AND FUTURE PROSPECTS

In this paper, the Genetic Algorithm (GA) is compared with the Social Spider Algorithm (SSA) for solving Travelling Salesman Problem (TSP). Standard benchmark instances were used to test the quality of the solutions and efficiencies of the two algorithms. The results show that the Social Spider Algorithm could also be appropriate in terms of competitive quality of solutions at less computational time than the Genetic Algorithm particularly for large problem instances. On the other hand, the Genetic Algorithm shows quite good convergence performance by multiple runs as a computational tool. The implications of these results are that swarm-based metaheuristic methods like SSA can be effective substitutes for the conventional evolutionary algorithms to address combinatorial optimization problems like the TSP. However, in some cases SSA can be prematurely converged so, taking into consideration the parameter tuning strategy and hybrid optimization techniques can be beneficial to make the algorithm more robust.

Potential future aims include counteracting the limitation of convergence of SSA and providing applications in other complex optimization problems and real-world problems.

ACKNOWLEDGEMENT

It is important to recognize the efforts of our supervisor who introduced us to the idea of Nature inspired Algorithms, and Metaheuristic Optimization Methods and the unconditional support and guidance throughout this project we also acknowledge the facilities and opportunities our Departments and Universities provided. Finally, we extend our appreciation to everyone who have directly or indirectly contributed to our research.

REFERENCES

[1] J. J. Q. James and V. O. Li, "A social spider algorithm for global optimization," Applied Soft Computing, vol. 30, pp. 614-627, May 2015.

[2] D. Whitley, "A genetic algorithm tutorial," Statistics and Computing, vol. 4, no. 2, pp. 65-85, June 1994.

[3] E. Cuevas, et al., "A swarm optimization algorithm inspired in the behavior of the social-spider," Expert Systems with Applications, vol. 40, no. 16, pp. 6374-6384, Nov. 2013.

[4] R. F. Foelix, Biology of Spiders, 2nd ed. New York, NY, USA: Oxford University Press, 1996.

[5] E. Bonabeau, M. Dorigo, and G. Theraulaz, Swarm Intelligence: From Natural to Artificial Systems. New York, NY, USA: Oxford University Press, 1999.

[6] F. Fernandez-Campon, "Group foraging in the colonial spider Parawixia bistriata," Animal Behaviour, vol. 74, no. 5, pp. 1551-1562, Nov. 2007.

[7] University of Waterloo, "National Traveling Salesman Problems," [Online]. Available: https://www.math.uwaterloo.ca/tsp/world/countries.html. [Accessed: Jan. 30, 2026].

[8] E.-G. Talbi, Metaheuristics: From Design to Implementation. Hoboken, NJ, USA: Wiley, 2009.

[9] R. S. Parpinelli and H. S. Lopes, "New inspirations in swarm intelligence: a survey," Int. J. Bio-Inspired Computation, vol. 3, no. 1, pp. 1-16, Jan. 2011.

[10] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical Report TR06, Erciyes Univ., 2005.

[11] X.-S. Yang, Engineering Optimization: An Introduction with Metaheuristic Applications. Hoboken, NJ, USA: Wiley, 2010.

[12] Y. Li, X. Zhang, and J. Wang, "Discrete Social Spider Algorithm for the Traveling Salesman Problem," in Proc. World Scientific Congress, 2021.

[13] M. Singh and R. Kaur, "Self-adaptive Social Spider Algorithm for Traveling Salesman Problem," Trans. Emerg. Telecommun. Technol., vol. 12, no. 7, pp. 3577-3588, 2021.
[14] H. Chen, L. Zhao, and Y. Sun, "Parallel Social Spider Algorithm based on population mining," Applied Soft Computing, vol. 128, Art. no. 109438, Oct. 2022.
[15] A. Gupta, P. Sharma, and S. Verma, "Improved Spider Optimization Algorithm," Scientific Reports, vol. 12, Art. no. 15632, 2022.
[16] K. R. Patil and S. S. Deshmukh, "Discrete Spider Monkey Optimization Algorithm for the TSP," Applied Soft Computing, vol. 89, Art. no. 106076, Apr. 2020.
[17] J. Brown and T. Wilson, "Comparative Analysis of Metaheuristic Algorithms for the TSP," Int. J. Sci. World, vol. 3, no. 2, pp. 45-58, 2025.
[18] L. Ahmed and M. Khan, "Review of Metaheuristics for TSP-based optimization," Comput. Ind. Eng., vol. 175, Art. no. 108872, Jan. 2023.
[19] R. Singh, P. Kumar, and A. Sharma, "Metaheuristic taxonomy and applications," Artif. Intell. Rev., vol. 56, pp. 1123-1160, Feb. 2023.
[20] F. Al-Mansoori and H. El-Sayed, "Discrete Artificial Algae Algorithm for the TSP," Int. Arab J. Inf. Technol., vol. 17, no. 4, pp. 567-574, July 2020.
[21] X. Li, Y. Zhao, and Z. Wang, "Solving TSP using IDINFO algorithm," ISPRS Int. J. Geo-Inf., vol. 14, no. 2, Art. no. 42, Feb. 2025.
[22] Y. Yang and X. He, "Self-tuning Firefly Algorithm," arXiv preprint arXiv:1610.08222, Oct. 2016.
[23] V. Kool, H. van Hoof, and M. Welling, "Graph Convolutional Networks for the TSP," arXiv preprint arXiv:1906.01227, June 2019.
[24] R. Patel and S. Mehta, "Survey of Metaheuristic Algorithms 2019-2024," arXiv preprint arXiv:2501.14769, Jan. 2025.
[25] P. Singh and M. Sharma, "Social Spider Optimization Algorithm: Theory and Applications," Int. J. Innov. Technol. Explor. Eng., vol. 8, no. 10, Aug. 2019.
[26] L. Zhao, Y. Sun, and H. Chen, "Salp Swarm Optimization: A Critical Review," arXiv preprint arXiv:2106.01900, June 2021.
[27] S. Kumar and R. Jain, "Improved Social Spider Algorithm variants," ICONTECH J. Eng. Res., vol. 5, no. 2, 2021.
[28] D. Applegate, et al., "Concorde TSP Solver," University of Waterloo, 1997.
[29] M. Held and R. Karp, "A dynamic programming approach to sequencing problems," J. ACM, vol. 9, no. 1, pp. 61-74, Jan. 1962.
[30] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," Oper. Res., vol. 6, no. 5, pp. 791-812, Oct. 1958.
[31] H. Chen, Y. Zhang, and S. Li, "Parallel swarm intelligence for large-scale optimization problems," J. Supercomput., vol. 78, pp. 1423-1445, 2022.

Submitted: 30th Jan 2026; Accepted: 14th April 2026;