Solving the Knight's Tour Problem using Swarm-based Meta-Heuristics

*An evaluation of different swarm-based meta-heuristics to solve the knight's tour problem

BS Steyn
(Department of Computer Science)
Stellenbosch University
Stellenbosch, South Africa
21740178@sun.ac.za

***Abstract*—This study evaluates swarm-based meta heuristics for solving the Knight's Tour Problem (KTP), comparing three Ant System (AS) variants, Ant Colony System (ACS), and Binary Particle Swarm Optimization (BPSO). Results show that ant-based algorithms achieve a 100% success rate, while BPSO fails to find a valid tour. ACS demonstrates superior convergence and scalability, maintaining performance across larger board sizes. Statistical analysis confirms significant differences in success rate, convergence speed, and variability. The findings highlight the effectiveness of reinforcement-based search for constrained optimization problems.**

I. Introduction

The Knight's Tour Problem (KTP)[5] is a combinatorial optimization and graph traversal problem, where a knight must visit every square on an N×N chessboard exactly once. Due to its exponential search space, KTP is classified as NP-hard, making exact solutions computationally infeasible for large board sizes.

This study evaluates the performance of swarm-based meta-heuristic algorithms—specifically, Ant System (AS) [2], Ant Colony System (ACS)[1], and Binary Particle Swarm Optimization (BPSO) [6]—in solving KTP. These algorithms are assessed based on success rate, computational efficiency, scalability, convergence behavior, and variability.

The results demonstrate that ant-based algorithms consistently find valid knight's tours, while BPSO struggles due to its inability to enforce move constraints. Statistical analysis confirms significant differences in convergence speed, solution variability, and overall performance. The findings highlight the effectiveness of pheromone-based solution construction for constrained pathfinding problems, with ACS emerging as the most scalable and reliable approach.

II. Background

This section provides background on the individual algorithms and the problem being solved

A. Knights Tour Problem (KTP)

The Knight's Tour Problem (KTP)[5] is a classical combinatorial optimization and graph traversal problem where a knight must visit every square on an N×N chessboard exactly once. The tour can be either open (the knight does not return to its starting position) or closed (the knight ends on its initial square).

Finding a complete knight's tour is NP-hard due to the exponential growth in possible move sequences as N increases[7]. The problem is commonly represented as a graph, where each square is a node, and legal knight moves define the edges. The objective is to construct a Hamiltonian path through this graph. Traditional brute-force approaches are computationally infeasible for large board sizes, necessitating the use of meta heuristic optimization techniques.

B. Ant Systems (AS)

The Ant System (AS)[2] is a swarm intelligence algorithm inspired by the foraging behavior of ants. It utilizes pheromone trails to reinforce paths taken by previous solutions, allowing the algorithm to learn promising moves over time.

Each artificial ant constructs a knight's tour by probabilistically selecting the next move based on pheromone levels (τ) and heuristic information (η), defined as:

$$P_{ij} = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{k \in \mathcal{N}_i} [\tau_{ik}]^\alpha [\eta_{ik}]^\beta} \tag{1}$$

where α and β control the influence of pheromone trails and heuristic information, respectively.

AS updates pheromones after each iteration, reinforcing all completed solutions, which can lead to slower convergence as the problem size increases. This limitation is addressed by more advanced ant-based algorithms.

C. Ant Colony System (ACS)

The Ant Colony System (ACS)[1] is an enhanced version of AS, improving both solution quality and convergence speed through three key modifications:

1) **Greedy State Transition Rule** – With probability q_0, an ant selects the best-known move rather than relying purely on probabilistic selection.

2) **Local Pheromone Update** – Pheromone levels are temporarily reduced after each move to encourage exploration.
3) **Global Pheromone Update** – Only the best tour found so far is reinforced, accelerating convergence.

These improvements make ACS more scalable and efficient, allowing it to maintain high success rates even as board size increases.

D. Binary PSO (BPSO)

Binary Particle Swarm Optimization (BPSO)[6] is an adaptation of Particle Swarm Optimization (PSO) for discrete spaces. Instead of continuous position updates, BPSO uses a probabilistic transfer function to determine if a move should be selected:

$$S(v_{i,j}^{(t)}) = \frac{1}{1 + e^{-v_{i,j}^{(t)}}} \tag{2}$$

where $v_{i,j}^{(t)}$ is the particle's velocity, influencing the probability of selecting a given move[3]. The position update rule is:

$$x_{i,j}^{(t+1)} = \begin{cases} 1, & \text{if } r < S(v_{i,j}^{(t)}) \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

where $r \sim U(0,1)$ is a random value ensuring stochastic movement.

Unlike AS and ACS

III. Evaluation Metrics

The probabilistic state transition for the AS algorithms, the stochastic state transition for the ACS algorithm and the stochastic parameters r_1 and r_2 lead to all the algorithms used in this paper to be non-deterministic processes. The same starting configuration may lead to different results. To counter the non-deterministic nature of the algorithms we run every test 30 times. The results of each test will be averaged. The performance metrics used in this paper are listed below:

1) Success rate
2) Computational efficiency
3) Scalability
4) Convergence Behaviour
5) Variability

The merits of the metrics are given below:

- *Success rate* represents the percentage of runs that results in the algorithm finding a complete knight's tour. The merit of this metric is to have a clear metric to track the ability of the algorithm to reliably solve the problem.
- *Computational efficiency* represents the time taken and number of iterations required to find a solution. The merit of this metric is to have a clear metric to determine the practical usability of these algorithms.
- *Scalability* represents the ability of the algorithm to maintain performance as the problem size increases. The merit of this metric is to have a clear view of the algorithmic adaptability to determine if the performance degrades drastically or slowly as the size increases.
- *Convergence Behaviour* represents the metric used to track the exploratory characteristic of each algorithm. The merits of this metric is to have a clear metric to determine the balance of exploration and exploitation for the various algorithms.
- *Variability* represents the how consistent each algorithm performs over multiple independent runs. The merits of this metric is to have a clear view of degree to which each algorithm varies per run.

The above metrics are evaluated at every iteration and averaged across all the iterations. A single run will consist of 5000 iterations or end if a complete knights tour has been found. The intention is to gather data on the effect of variations to the parameters of each algorithm.

IV. Evaluation Procedure

This section provides insight into the empirical process used to compare the performance of the various algorithms used in this paper. The testing was performed on the Knights Tour Problem[5].

A. Algorithm Parameter Optimisation

The performance of AS, ACS and BPSO are highly dependent of the choice of algorithmic parameters. Poor choices for these parameters may lead to slow convergence, suboptimal solutions or early stagnation. Optimisation of these parameters is vital for the exploration vs exploitation balance of the algorithms.

1) AS parameters: The parameters that were optimised for the AS algorithms are given below:

- α: The higher the value of α the more exploitation is favoured where as lower values favour exploration. The range tested for was [0.5-2.5].
- β: The higher the value of β favour greedy moves based on immediate distance. If the value is too high it may lead to being trapped in local optima. The range tested for was [1-5].
- ρ: The higher the value of ρ the more exploration is favoured where as lower values favour previous values. The range tested for was [0.1-0.9].
- Q: Higher values lead to exploitation be favoured. The range tested for was [1-50].
- τ_0: Low values lead to early exploration where as high values lead to early exploitation. The range tested for was [0.001-0.1].

2) ACS parameters: The parameters that were optimised for the ACS algorithm are given below:

- α: The higher the value of α the more exploitation is favoured where as lower values favour exploration. The range tested for was [0.5-2.0].
- β: The higher the value of β favour greedy moves based on immediate distance. If the value is too high it may

lead to being trapped in local optima. The range tested for was [1-5].
- ρ: The higher the value of ρ the more exploration is favoured where as lower values favour previous values. The range tested for was [0.1-0.9].
- q_0: High values force greedy selection where as low values encourage probabilistic selection. The range tested for was [0.5-0.99].
- τ_0: Low values lead to early exploration where as high values lead to early exploitation. The range tested for was [0.001-0.1].

3) BPSO parameters: The parameters that were optimised for the BPSO algorithm are given below:

- w: Higher values favour exploration where as lower values favour exploitation. The range tested for was [0.4-0.9].
- c_1: Determines the effect personal best solution has on the iteration. The range tested for was [1.0-2.5].
- c_2: Determines the effect of the neighbourhood best has on the iteration. The range tested for was [1.0-2.5].

The optimal values for the parameters listed above for AS, ACS and BPSO by using the grid search method where all combinations of the parameters are tested and the one with the highest success rate and lowest iterations on average over 30 runs is selected.

B. Algorithm Parameter Optimisation Results

The optimised values for the AS-D, AS-C, AS-Q, ACS and BPSO can be seen in their representitive table.

TABLE I: Table showing the optimised values for the AS-D algorithm

Parameter	Value
α	1.0
β	3.0
ρ	0.6
Q	41.0
τ_0	0.061

TABLE II: Table showing the optimised values for the AS-C algorithm

Parameter	Value
α	2.5
β	3.0
ρ	0.1
Q	21
τ_0	0.091

TABLE III: Table showing the optimised values for the AS-Q algorithm

Parameter	Value
α	1.5
β	3.0
ρ	0.4
Q	1.0
τ_0	0.001

TABLE IV: Table showing the optimised values for the ACS algorithm

Parameter	Value
α	0.5
β	1.0
ρ	0.1
q_0	0.7999
τ_0	0.061

TABLE V: Table showing the optimised values for the BPSO algorithm

Parameter	Value
w	0.6
c_1	1.8
c_2	1.8

The optimised AS, ACS and BPSO algorithms are now evaluated using the metrics discussed previously. Statistical tests are then performed to determine if there is a significant difference in the results for the various algorithms.

V. Results

This section presents the empirical results and observations of the various algorithms discussed in this paper. These results are obtained after applying the metrics discussed in the Evaluation Metrics section.

A. Success Rate

The most basic metric used to determine whether an algorithm can solve the KTP. Consider the table:

TABLE VI: Table showing the success rate at board size 8x8 for all algorithms

Algorithm	Success Rate
$AS-C$	100.0%
$AS-D$	100.0%
$AS-Q$	100.0%
ACS	100.0%
$BPSO$	0.0%

Table VI clearly shows that at the size of a standard chessboard the success rate of the ant algorithms are identical. The table also shows that BPSO is not able to find a valid tour. This likely due to BPSO not enforcing valid knight moves, leading to infeasible solutions where as AS and ACS construct solutions incrementally ensuring each move is feasible. To further examine the AS and ACS algorithms the scalability of the success rate is examined. Consider the graph below:

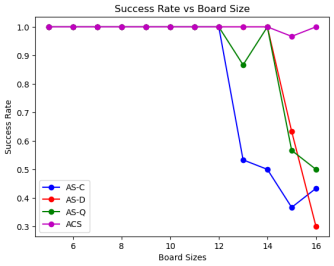

Fig. 1: Figure showing how each algorithms success rate scales

B. Convergence Behaviour

A effective measure of how a algorithm is the speed with which it can locate the solution. Consider the table:

TABLE VII: Table showing the average number of steps needed to find a solution at board size 8x8 for all algorithms

Algorithm	Iteration to find first solution
$AS-C$	1.0
$AS-D$	4.0
$AS-Q$	0.4333
ACS	0.0
$BPSO$	0.0

Table VII clearly shows that some algorithms converge faster than others. Performing the Friedman test shows that there is a statistical difference between the algorithm. The scalability of the success rate is further examined for AS and ACS.. Consider the graph below:

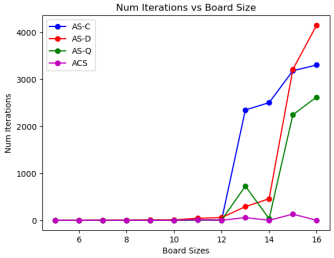

Fig. 2: Figure showing how each algorithms required iterations scales

C. Variability

Variance is a crucial statistical metric to measure the dispersion and consistency of data points in a dataset. Consider the table:

TABLE VIII: Table showing the variance at board size 8x8 for all algorithms

Algorithm	Variance
$AS-C$	0.44556
$AS-D$	8.4489
$AS-Q$	0.6489
ACS	0.1289
$BPSO$	0.0

Table VIII clearly shows that some algorithms have a higher variance than other. To determine whether this is statistically significant we first perform a Friedman test to determine if the difference is significant between the top three consistant algorithms. BPSO is being ignored as the reason it has zero variance is that it did not find a complete route once within the 30 iterations. The Friedman test with a α value of 0.05 shows that there is a significant difference. The Friedman test resulted in p = 0.0022 which is below the α = 0.05 threshold, indicating statistical significance To further examine the AS and ACS algorithms the scalability of the variance is examined. Consider the graph below:

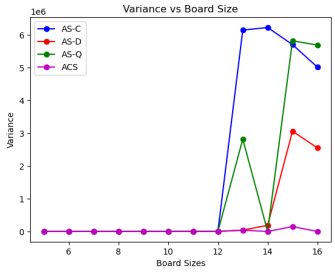

Fig. 3: Figure showing how each algorithms success rate scales

D. Scalability

From Figure 1, 2 and 3 it is clear to see that ACS is the only algorithm that scales well. Since BPSO was only able to consistently find solutions for a board size of 5×5, it has been excluded from the figures. Of the remaining algorithms AS-D scaling has the least effect on AS-D's variance where as AS-D's success rate and iterations required scale the worst. AS-C is normally the first algorithm to suffer from scaling problems however it does not reach the extreme scaling effect of AS-D and AS-Q. ACS is more resistant to scaling due the to fundemental difference in the pheromone update mechanisms, exploration strategies and convergence behaviour between the algorithms.

E. Rankings of the Algorithms

Consider the table:

TABLE IX: Table showing the variance at board size 8x8 for all algorithms

Algorithm	Success Rate	Iteration Required	Variance	Overall Rank
$AS-C$	1	3	2	2
$AS-D$	1	4	4	3
$AS-Q$	1	2	3	2
ACS	1	1	1	1
$BPSO$	5	5	5	5

Table IX demonstrates that ACS is the best algorithm from the algorithms examined in this paper.

VI. Conclusion

This study evaluated the performance of three Ant System (AS) variants, the Ant Colony System (ACS), and Binary Particle Swarm Optimization (BPSO) in solving the Knight's Tour Problem (KTP). The evaluation was based on five performance metrics: success rate, computational efficiency, scalability, convergence behaviour, and variability.

The results demonstrated that the ant-based algorithms consistently found complete knight's tours, achieving a 100% success rate on an 8×8 chessboard, whereas BPSO failed to find a valid solution. The performance disparity is attributed to BPSO's inability to enforce valid knight moves, leading to infeasible solutions.

Analysis of convergence behaviour indicated that ACS outperformed all AS variants, requiring fewer iterations to locate a valid knight's tour. The scalability analysis revealed that ACS maintained high success rates across larger board sizes, whereas the AS variants exhibited performance degradation as problem size increased. This result aligns with the structural differences in pheromone update mechanisms and exploration-exploitation balance between AS and ACS.

Statistical analysis confirmed that the differences in success rate, convergence speed, and solution variability between the algorithms were statistically significant. The Friedman test was used to verify the significance of these differences, showing that ACS consistently ranked as the best-performing algorithm across all performance metrics.

The findings highlight the importance of algorithm design choices, particularly in reinforcement strategies and solution construction mechanisms, when applying swarm-based meta heuristics to constrained path-finding problems. Future work should explore hybrid approaches that integrate local optimization strategies with swarm intelligence to enhance scalability and robustness in solving the Knight's Tour Problem.

References

[1] Marco Dorigo and Luca Maria Gambardella. "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem". In: *IEEE Transactions on Evolutionary Computation* 1.1 (1997), pp. 53–66. DOI: 10.1109/4235.585892.

[2] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. "The Ant System: Optimization by a Colony of Cooperating Agents". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1 (1996), pp. 29–41. DOI: 10.1109/3477.484436.

[3] Russell Eberhart and Yuhui Shi. "Particle Swarm Optimization: Developments, Applications and Resources". In: *Proceedings of the IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2001, pp. 81–86. DOI: 10.1109/CEC.2001.934374.

[4] Philip Hingston and Graham Kendall. "Enumerating Knight's Tours Using an Ant Colony Algorithm". In: *Proceedings of the Congress on Evolutionary Computation (CEC)*. IEEE, 2004, pp. 2293–2298. DOI: 10.1109/CEC.2004.1331173.

[5] M. Ismail, Ezreen Farina Shair, Arfah Syahida Mohd Nor, et al. "A Preliminary Study on Solving Knight's Tour Problem Using Binary Magnetic Optimization Algorithm". In: July 2013.

[6] James Kennedy and Russell C. Eberhart. "A Discrete Binary Version of the Particle Swarm Algorithm". In: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 1997, pp. 4104–4108. DOI: 10.1109/ICSMC.1997.637339.

[7] Allen J. Schwenk. "Which Rectangular Chessboards Have a Knight's Tour?" In: *Mathematics Magazine* 64.5 (1991), pp. 325–332. DOI: 10.1080/0025570X.1991.11996309.