Ant Colony Optimisation (ACO) and its performance solving the security van problem (knapsack 0/1)

Michael Dean
University of Exeter
md668@exeter.ac.uk

ABSTRACT

In this paper we design an ant colony optimisation algorithm to effectively pack a security van full of money bags optimising the monetary value within a fixed weight requirement, commonly referred to as a knapsack 0/1 problem. We also compare 3 other nature-inspired algorithms areas to the knapsack 0/1 problem.

KEYWORDS

Ant colony optimisation, swarm intelligence, nature-inspired computation

1. LITERATURE REVIEW

1.1 Binary Particle Swarm Optimisation (BPSO) Algorithms

Particle swarm optimisation (PSO) proposed by Kennedy and Eberhart [1] is inspired by bird flocks and their ability to adapt their flight based on both their own flight experience and that of the collective. They designed an algorithm combined of multiple particles in a swarm and updated the velocities of these particles based on an attraction to the current best solution in the swarm and an attraction to the particles local best solution. Whilst initially designed to solve problems with continuous solutions, Kennedy and Eberhart [2] proposed another version of PSO in 1997 called BPSO specifically to address problems with discrete solutions, modifying the velocity to represent a probability of the object taking a binary position. Since the creation of the original BPSO algorithm, many modified forms of BPSO have been made such as Quantum Inspired BPSO (QBPSO) [3] taking inspiration from quantum mechanics and replacing particle velocity with displacement. Traditional QPBSO is well known to have limitations in discrete problems, such as being slow to converge and often getting stuck in local optima, however Li et al [4] implemented a quantum-based BPSO method called IBQPSO, an improved binary quantum-behaved PSO incorporating some mutation techniques to help deal with local optima and a new repair and fitness method to help improve infeasible solutions. It was tested specifically on knapsack 0/1 problems and outperformed popular BPSO algorithms on the such as BQPSO and BPSO and some evolutionary algorithms such as NbinDE, T-NBDE, and BAO-T algorithms in small and medium scaled problems. IQBPSO appears to take the best elements of both a genetic algorithm approach and PSO. However, like with many modified nature-inspired algorithms for the knapsack 0/1, the reliance on a repair function to generate feasible solutions is inconvenient. Algorithms such as ACO do not need a repair function just termination criterion when creating a path. An ACO approach will never generate an infeasible solution if it is terminated correctly. If an infeasible solution is created in IBQPSO, there is a multiple stage repair function highlighted involving potentially very computationally complex processes such as sorting and searching, giving the worst-case time complexity component of the repair function as O(mn), where m is the population size and n is the number of items in the knapsack. A comparison between ACO and IBQPSO would need to be performed for a more detailed insight, however I believe its unlikely to match the performance of ACO.

1.2 Genetic Algorithms (GA)

Genetic Algorithms are widely studied and have shown their suitability for deployment on knapsack 0/1 problems. GA algorithms mimic evolution by generating offspring from parents, performing mutations and assessing the fitness of new solutions to determine who should remain in the population. Often deployed with repair functions to convert infeasible solutions to feasible solutions, they can search for the global optimum unlike some comparable algorithms such as the greedy algorithm, which can only select the locally optimal bag at each stage in a knapsack algorithm. He et al [6] implemented a solution to knapsack 0/1 using an evolutionary algorithm with multiple objectives called MOGA, by generating helper functions i.e more fitness functions to maximise elements such as the number of items in the knapsack and the value to weight ratio. Resultant performance showed improvements over the non-genetic greedy algorithm and over a mixed strategy evolutionary algorithm that includes a probabilistic repair function. One limitation of the study is the lack of break-down of computational complexity surrounding the novel added helper functions, adding these additional fitness functions converts the knapsack 0/1 problem from being single objective to multi-objective, significantly increasing the number of calculations per evaluation of the algorithm and thus must increase the overall runtime and memory utilisation. Although MOGA is slightly different to a traditional GA and has not been directly compared to an ACO approach, it likely has many similar advantages and disadvantages to an ACO approach for a knapsack 0/1 problem. Whilst a GA approach can converge too quickly if diversity is lost too fast, an ACO approach can converge too quickly if too much pheromone is deposited early on. GAs can also present challenges for a knapsack 0/1 problem as the mutations can create infeasible results that need to be refined, sometimes through multiple stage repair functions. In an ACO approach, when an ant explores over the weight limit, the path can be truncated by removing the last node to produce a feasible solution.

1.3 Improved Cuckoo Search (ICS) Algorithm

Cuckoo search (CS) was proposed by Yang and Deb, replicating the behaviour of a cuckoo to invasively lay their eggs in the nests of other birds. Yanhong Feng et al [8] proposed improvements to the original cuckoo search by first converting the algorithm into the discrete state and then using new strategies such as an adaptive step size, greedy transform method and genetic mutation operator called improved cuckoo search (ICS). The implementation of a genetic operator should help avoid local optima, whilst the greedy transform method should improve the decision-making of the algorithm**.** The results show improvements in small-medium knapsack problems. with ICS showing improvements over CS and harmony search (HS) another metaheuristic algorithm. ICS performs marginally worse than the other algorithms in large knapsack 0/1 problems, however the differences between ICS, CS and HS seem marginal. The real improvement appears to be in the time to converge. A direct comparison between ACO and ICS would need to be performed to make solid conclusions, especially since timing performance cannot be compared due to ICS being run on a very slow old CPU. The abandonment of poor solutions could provide one advantage, as poor solutions would be removed instead of slowly decaying over iterations in an ACO approach.

2. METHOD

2.1 Design

An ACO algorithm was implemented to solve the security van packing problem. In this algorithm, a fixed number of ants would perform a set number of tours based on the desired number of fitness evaluations. At the end of each tour, each ant would deposit some pheromone on every edge of the graph that the ant traversed. The pheromone present and the heuristic value of that edge would influence the decision of other ants during following tours.

2.1.1 Graph Structure

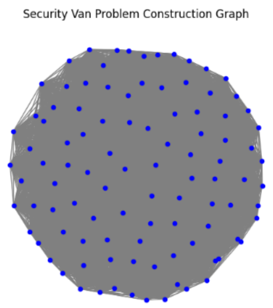

Figure 1: Security Van Problem Construction Graph

The graph used linked each money bag to every other money bag. Hence, from a starting position of any bag, an ant could travel to any other bag in the graph if it has not already visited that bag. The result is an extremely intricate graph shown in figure 1, where a bag (node) is shown in blue connected by 99 edges to each other bag in grey.

2.1.2 Heuristic Matrix

2.1.2.1 Heuristic Value

The heuristic value linking a node of the graph to another node was calculated using the new nodes money/weight ratio.

$$heuristic\ value_{ij} = \frac{money_j}{weight_j}$$

2.1.2.2 Defining visited nodes

A "visit" of a node in the graph was performed by zeroing the column for the visited node in the heuristic matrix, therefore when the visitation probability was calculated between any node and a visited node, the probability of the ant moving to a visited node would be zero.

2.1.3 Fitness Function

The fitness function to define the quality of a solution and thus how much pheromone would be deposited was defined as

Fitness = $m * \frac{current\ ant\ fitness}{population\ best\ fitness}$ where m = pheromone scaler

Hence the amount of pheromone deposited would be a normalised value between 0 and 1 multiplied by a scalar factor m. Where pheromone equal to 1*m would be deposited for the best solution in the population. Using a normalised pheromone deposit would prevent high pheromone deposits for poor solutions, such as 2000 pheromone deposited for a £2000 fitness solution.

2.2 Experimentation

To implement the algorithm effectively, defined constants need to be optimised to produce the best results. The main constants in an ACO implementation are the pheromone evaporation rate (e), the number of ants, the number of evaluations and any scalar value applied to the pheromone deposits. The effects of these parameters were identified by performing sweep tests.

2.2.1 Population Size

2.2.1.1 Small populations (p)

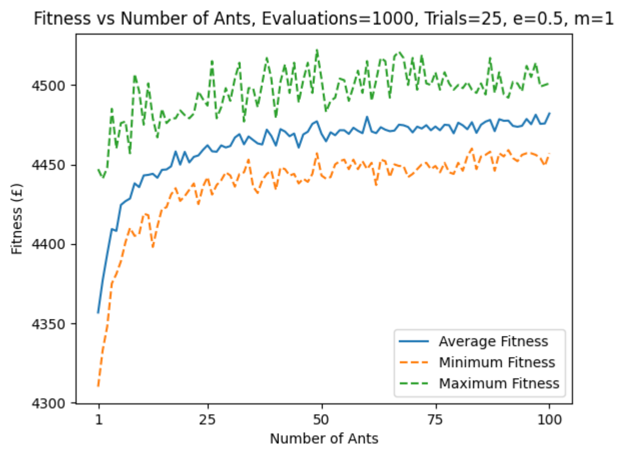

Figure 2: Fitness vs Number of Ants (Small)

2.2.1.2 Large populations (p)

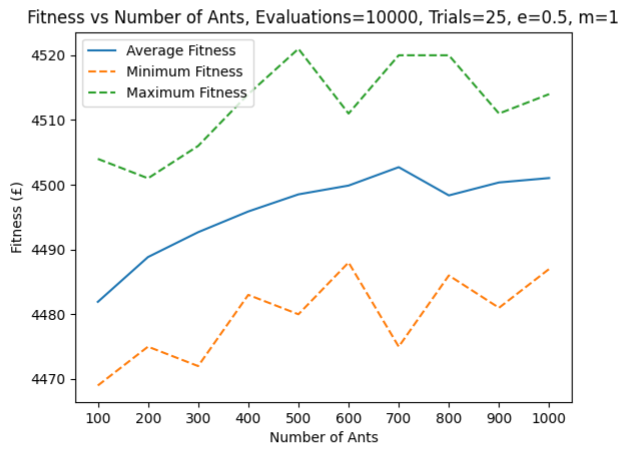

Figure 3: Fitness vs Number of Ants (Large)

2.2.2 Pheromone Evaporation Rate (e)

$$new\ pheromone = (1 - e) * old\ pheromone$$

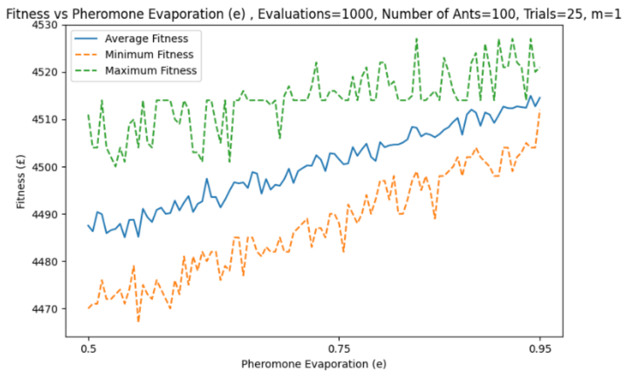

Figure 4: Fitness vs Pheromone Evaporation Rate (e)

2.2.3 Pheromone deposit scaler (m)

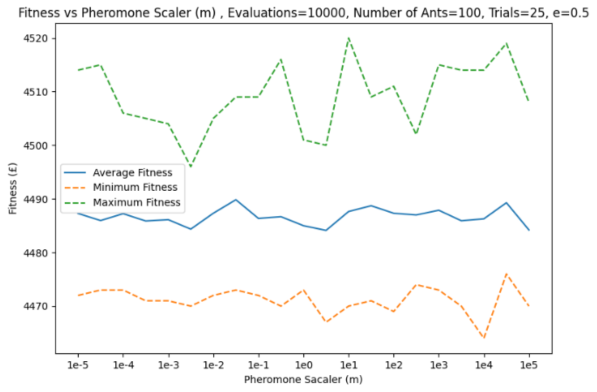

Figure 5: Fitness vs Pheromone Deposit Scaler (m)

2.3 Parameter Configuration

Based on the experiments conducted, a desirable parameter configuration was defined to maximise performance of the algorithm in a reasonable runtime.

Table 1: Algorithm Configuration

Parameter	**Value**
Pheromone Evaporation Rate (e)	0.95
Number of Ants (p)	500
Pheromone Scaler (m)	1
Number of Evaluations	25000

The proposed configuration had a runtime of ~14.16 s on an i7 14700k and was run over 5 trials to produce an average result.

2.4 Results

Table 2: Results by Trial

Trial Number	**Fitness**
1	4527
2	4521
3	4522
4	4522
5	4515

A summary of the results was produced in the table below

Table 3: Result Metrics

Result Type	**Value**
Minimum Fitness	4515
Average Fitness	4521.40
Maximum Fitness	4527
Fitness Standard Deviation	3.83
Fitness Variance	14.64

3. DISCUSSION AND FURTHER WORK

3.1 Influence of Parameter Experiments

3.1.1 Population Size (p)

3.1.1.1 Small Population

When evaluating small population sizes, only 1000 evaluations were done since the number of tours is equal to the number of evaluations / population size, hence running multiple trials over very small population sizes was very time consuming due to the large number of tours required. Additionally, with only each ants pathing being multi-threaded due to the need to wait for all ants to complete a tour to update the pheromone matrix, a suitable optimisation could not be introduced to speed up the small population sweeps. As shown in figure 2, increasing the population size from a singular ant to 25 ants created a substantial difference in fitness over 25 trials, with the improvement trailing off between 25-100 ants.

3.1.1.2 Large Population (p)

Large populations sizes 100 -> 1000 ants were evaluated over 10,000 fitness evaluations. As seen in figure 3**,** the trend of increased population size increasing fitness continues, however to a much more limited degree. Since the ant pathing code was multi-threaded and increasing the number of ants reduced the number of tours, a high population was desirable to allow for more evaluations to be performed in a reasonable length of time.

3.1.2 Pheromone Evaporation (e)

Pheromone evaporation is an important aspect of the ACO algorithm. Setting an evaporation rate too low does not allow the ants to fully explore the graph and tends to cause early convergence, which is not ideal. However, when pheromone evaporation is very high and the ant population is very low, quality solutions can evaporate very quickly. In figure 4, as the pheromone evaporation rate is increased, the minimum, maximum and average results begin to converge at the maximum result. This is a desirable property and was why the maximum pheromone evaporation rate was chosen for our ACO implementation.

3.1.3 *Pheromone Deposit Scaler (m)*

A pheromone deposit scaler was introduced to experiment with increasing and reducing the amount of pheromone deposited by each ant. In figure 5, this parameter was varied between 10^{-5} to 10^{5}. The results were not significant, and no reasonable assumption could be made that increasing or decreasing the pheromone deposit scaler significantly improved or reduced the fitness of the best solution, even after multiple 25 trial runs.

3.2 Further Experiments and Analysis

3.2.1 *Low evaluation pheromone deposit scaler*

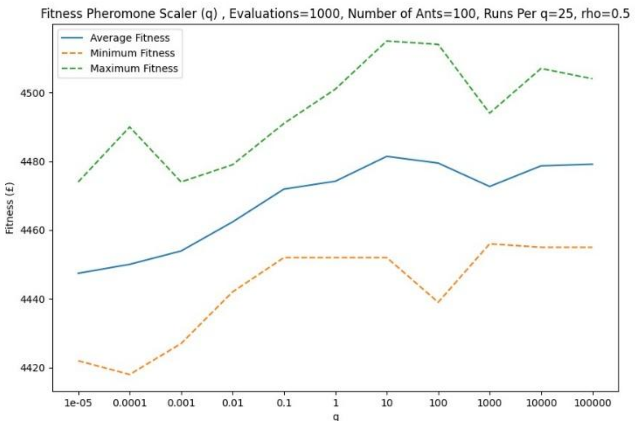

Figure 6: Fitness vs Pheromone Deposit Scaler (m) low evaluations

Where q =m, rho = e and runs per q = trials

There was evidence that increasing the pheromone deposit scaler at low numbers of evaluations improved the overall fitness of the best solution. However, this could not be replicated at the suggested 10,000 fitness evaluations to ensure statistical significance.

3.2.2 *Ant-Exploration Heatmap*

An ant exploration heatmap was produced to highlight nodes that were the most visited over multiple trials. This heatmap can then be compared to a map of the nodes with the highest money/weight ratio to correlate how an ACO approach differs from heuristic implications.

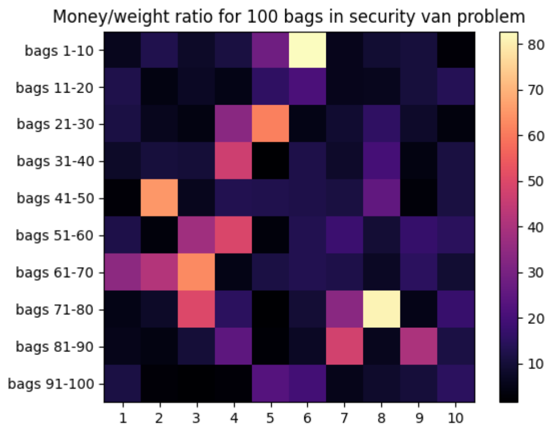

Figure 7: Money/weight ratio for all 100 bags in the algorithm

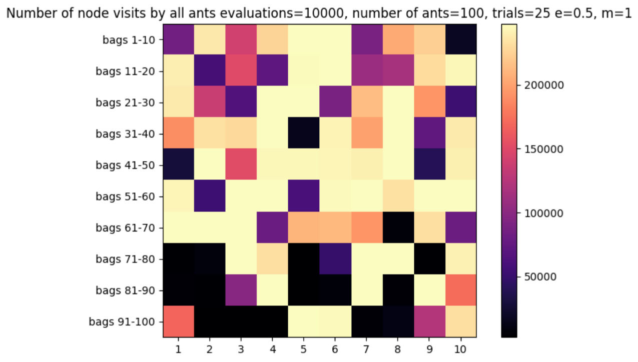

Figure 8: Ant Exploration Heatmap

Contrasting figure 7 with a heatmap of the most visited nodes in 25 trials shown in figure 8 shows a very interesting correlation. Some clear similarities can be seen in bags with the worst money to weight ratio, for example bags 71, 72, 81, 82, 92, 93, 94 and with bags 75, 76, 85, 86 perfectly map as both low money to weight ratio and low visitation by the ants. However, many low to medium money to weight ratio bags are very highly explored due to the relatively generous weight limit of the security van.

3.3 Future Improvements

3.3.1 *Modified Heuristic Matrix*

Our implementation of the heuristic matrix did not include any information about the starting node from which the ant was travelling from, or in other words, it was entirely dependent on the money/weight ratio of the new bag. Upon reflection, a heuristic function that included characteristics from both the start bag and the new bag may have provided some benefit in increasing the probability of choosing a bag that fit well with the bag the ant had just added to the security van. Additionally, since the average between a bag with a high money to weight ratio and a bag with a low money to weight ratio would meet somewhere in the middle, it’s possible it could encourage the ant to explore bags that would be probabilistically overlooked in our implementation.

The decision to use the heuristic matrix to remove nodes that had already been visited was not the most efficient way to implement this feature. This decision required each ant to have its own heuristic matrix, which for large populations requires more memory. A better solution would have been to introduce a list for each ant containing all nodes and pop each node out of the list as its visited. This way probabilities would only be calculated for the nodes that are left inside the list, and all ants could rely on a shared heuristic matrix, thereby the amount of memory and operations required.

3.3.2 *Hybrid Pheromone Evaporation*

Whilst the pheromone evaporation rate set high provided a significant improvement to results, it is likely a hybrid pheromone evaporation configuration could produce better results in a more complex problem with many nodes. This entails using a high pheromone evaporation rate at the beginning of the algorithm to encourage exploration, and decreasing the pheromone evaporation rate as the number of tours increases to encourage convergence.

3.4 Conclusion

The results achieved by the ACO were excellent. The maximum result achieved over five trials was very high whilst maintaining a fast run-time. The results achieved are the best the algorithm can obtain, increasing the number of evaluations further proved to not yield any higher fitness solutions.

3.4.1 *Comparison to other nature-inspired algorithms*

Whilst conclusions are hard to draw without specifically testing each nature-inspired algorithm on this knapsack 0/1 variant, it’s likely that ACO is the best performing algorithm for this task. Most nature-inspired algorithms are designed to solve problems with continuous solutions and need transfer functions and repair functions to generate feasible discrete solutions. ACO is well proven to perform extremely well on discrete problems such as the TSP (travelling salesman problem) which shares similarities to this problem.

4. REFERENCES

[1] James Kenedy and Russel Eberhart (1995) Particle Swarm Optimization. *Proceedings of the IEEE International Conference* on Neural Networks, pp 1942-1948. http://dx.doi.org/10.1109/ICNN.1995.488968

[2] James Kenedy and Russel Eberhart (1997). A discrete binary version of the particle swarm algorithm. *Proceedings of IEEE International Conference on Systems, Man, and Cybernetics* (Vol. 5, pp. 4104-4108) https://doi.org/10.1109/ICSMC.1997.637339

[3] Jun Sun, Wei Fang, Xiaojun Wu, Vasile Palade and Wenbo Xu. Quantum-behaved particle swarm optimization: analysis of individual particle behavior and parameter selection. *Evol Comput 2012 Fall* https://doi.org/10.1162/evco_a_00049

[4] Xiaotong Li, Wei Fang, Shuwei Zhu. (2023) An improved binary quantum-behaved particle for swarm optimization algorithm for knapsack problems. *Information Sciences* (Vol. 648, 119529) https://doi.org/10.1016/j.ins.2023.119529

[5] Yun-Won Jeong, Jong-Bae Park, Se-Hwan Jang, Kwang Y. Lee (2009) A New Quantum-Inspired Binary PSO for Thermal Unit Commitment Problems. *15th International Conference on Intelligent System Applications to Power Systems* (pp.1-6) *https://doi.org/10.1109/ISAP.2009.5352869*

[6] Jun He, Feidun He and Hongbin Dong. (2014) A Novel Genetic Algorithm using Helper Objectives for the 0-1 Knapsack Problem https://arxiv.org/pdf/1404.0868

[7] Xin-She Yang and Suash Deb. (2009) Cuckoo Search via Lévy flights. *World Congress on Nature & Biologically Inspired Computing (NaBIC)* https://doi.org/10.1109/NABIC.2009.5393690

[8] Yanhong Feng, Ke Jia and Yichao He. (2014) An improved hybrid encoding cuckoo search algorithm for 0-1 knapsack problems. *Computer Intelligence and Neuroscience (2014) https://doi.org/10.1155/2014/970456*