# ECMM409-Coursework
# Parameter Affect on ACO performance

**Author**
740009526

## 1 Literature Review

### 1.1 What is Ant Colony Optimization?

ACO is a meta-heuristic that mimics the way ants gather food to optimize a path through a graph. The key to the algorithm's success is fourfold [5]:

- An iterative, distributed search of the graph, through the use of multiple "ants"
- Stochastic node selection in the search, through the use of some heuristic to deploy "pheromones" on better edges in the graph, as to bias better paths
- The balance of exploration in the early stages and exploitation in the later stages, without the need for self-adaption, due to the $\alpha$ and $\beta$ parameters in the node selection.
- The ability to escape sub-optimal solutions in multi-model fitness landscapes, unlike simpler heuristics, such as hill climbing or gradient decent.

ACO is commonly used to approximate solutions to NP-Hard problems, one of many nature-inspired algorithms for tackling such problems[2]. The focus of this paper is on the 0-1 knapsack problem, the following sub-sections will define the knapsack sack problem and review alternative nature-inspired approach to finding optimal solutions.

### 1.2 Nature Inspired Algorithms Applied to the 0-1 Knapsack Problem

The 0-1 Knapsack Problem (KP01) is a combinatorial optimization problem were $N$ objects of value $p_i$ and $w_i$ must be grouped together under a constraint $C$, as to maximize the value within the constraint [7]. Its formulated as such below, $X_i$ is a binary decision operator, as to whether the object is added or not.

$$\begin{gathered}\text{Max} \sum_{i=1}^{n} p_i \times X_i \\ \sum_{i=1}^{n} w_i \times X_i \leq C \\ X_i \in 0, 1\end{gathered} \tag{1}$$

#### 1.2.1 ACO Variations

Many adaptation of ACO have been developed in an attempt to find better solutions with less iterations[5]. Three well-known adaptions are covered here, Elitist Ant System, Rank-Based Ant System and Max-Min Ant System.

- **Elitist Ant System (EAS)** [9] works the same way as simple ACO, except applies additional pheromones on edges in the *best-so-far* tour.
- **Rank Based Ant System** [5] involves additional reinforcement to the *best-so-far* tour, as in EAS, but also deposits additional pheromones depending on value of an ant's tour rank proportional to all ants. Before pheromone is calculated, all tours are ranked and their pheromones additions weighted by said rank.
- **Max-Min Ant System (MMAS)** [5, 19] has four modifications to simple ACO.
  - To balance against rapid convergence, all possible $\tau$ values are limited to the range $[\tau_{min}, \tau_{max}]$. When a new *best-so-far* tour is found, $\tau_{max}$ is updated.
  - Rather than a uniform distribution of pheromones, all $\tau$ values are initialized at $\tau_{max}$.
  - The algorithm only allows the iteration's best or the *best-so-far* tour to deposit pheromones.

- Every time the *best-so-far* tour does not improve for a set number of iterations, pheromone levels are re-set to $\tau_{max}$.

#### 1.2.2 Annealing Algorithm (SA)

Annealing is a process for improving metals[3]. In annealing, metals are heated just below their melting point, allowing the metal's internal structure to change. The material is then cooled down, where at this lower temperature, better lattice structures from within the metal form. This process is repeated several times, resulting in a stronger metal.

SA replicates this process, the solution being the state of the material, the objective function as the energy of the material and optimal solutions being those with minimal energy[1, 5].
The search starts with an initial solution $s$ and randomly compares it to another, $s'$. If the objective function of $s'$ is larger, then the solution is accepted, however, if the value is worse it can still be accepted according to probability $T$, temperature. The search starts with a high $T$, encouraging high exploration, but as $T$ is decreased towards 0, exploitation of better solutions takes priority[16].

SA has similar traits to ACO when applied to KP01, steps in the search space are not limited by the proximity of solutions, solutions are chosen through a stochastic process that changes probability throughout the search, and both can escape sub-optimal solutions. SA has been applied to KP01 problems with much succuss with small and large problem sets[14].

#### 1.2.3 Multiobjective Evolutionary Algorithm (MOEA)

Evolutionary algorithms mimic the evolutionary principals of a population adapting to its environment[10]. This is usually done by defining an initial population, with each member of the population being a candidate solution with a fitness value. A selection process, often elitist, to provide *evolutionary* pressure to the population and finally crossover and mutation operators to stochastically modify the selected solutions. These modified solutions go on to form the next generation and the process repeats[18].

Multi-objective optimization algorithms attempt to find the set of solutions that, with respect to multiple objectives and thus multiple fitness functions, cannot be better in some objective without performance loss in another, called non-dominated solutions[8]. This set in objective space is referred to as the Pareto front. The quality of a multi-objective algorithm is measured by its ability to both find solutions that converge on the Pareto front but are also diverse in their coverage, ideally producing a uniform distribution of solution across the front.
Evolutionary algorithms are often used for such problems[11] for their distributed search and ability to balance both diversity and convergence, such as NSGA-II[4], NSGA-III[20] and MOEA/D[21]. There has also been extensive research into the handling of constraints in MOEA, in-which they excel at.

MOEA have been applied to the KP01 problem to great succuss when finding the Pareto-optimal solutions[12, 13]. However, problem modification needs to be made, such as adding an extra profit objective or converting the constrains into an objective to be minimized, although the latter is likely to result in poor solutions, the former scales well with more complex knapsack problems.

## 2 Methodology

The defining features of an ACO algorithm is the edge heuristic, the transition rule, pheromone updating, pheromone evaporation and the chosen parameters.

### 2.1 Heuristic

The heuristic $\eta$ determines the value of an ant traveling from one node to another in the graph. For example, when ACO is applied to the traveling salesman problem (TSP), $\eta$ is $\frac{1}{d_{ij}}$, where $d_{ij}$ is the distance between the two cities, $i, j$. However, a simple conversion cannot be applied to the knapsack problem, since the problem is not minimizing a value as in TTP but maximizing a value within a constraint. Taking the inverse of a node's value would punish valuable nodes and ignore the weight. A good $\eta$ must account for both value and weight. Only accounting for value might prioritize valuable bags with significantly large weights, and only accounting for weight might prioritize light bags with poor values.

The heuristic used was the value-weight ra-

tio of the node to move to, $\frac{v_i}{w_i}$. This method rewards larger value with smaller weights, while accounting for abnormal nodes that may have both significantly large values and weights.
The heuristic has been used successfully in applications of ACO to the problem before. Krzysztof Schiff[17] showed that the ratio, named *AKA3* in the paper, can find higher cost solutions faster than the alternative heuristics given,

- AKA1 - $\frac{\frac{z_j}{w_j}}{V_c}$
- AKA2 - $\frac{\frac{z_j}{w_j}}{C}$

where $w_j$ & $z_j$ are the cost and value of the selected node and $V_c$ & $C$ are the current tour capacity and the weight constraint.
Another study also examined the same heuristic, coined *MPW*. $\eta$ was also compared too two other heuristics, the maximum profit per object and the minimum weight per object in [6]. The authors found that MPW significantly outperformed the other two.

### 2.2 Pheromone Evaporation

Pheromone evaporation avoids a quick convergence to sub-optimal paths by reducing the level of pheromones on all paths, by encouraging exploration and effectively "forgetting" poor choices made by previous ants. Evaporation is applied to all by

$$\tau_{ij} \leftarrow (1-p)\tau_{ij}, \quad \forall(i,j) \in A$$

Pheromone evaporation is applied before pheromone addition rather than after, as proposed by M.Dorigo in S-ACO [5].

### 2.3 Pheromone Updating

The pheromone update$\tau$ formulae uses the heuristic's intuition, of balances an object's value and weight, to deposit pheromones proportional to the total value-weight ratio of each ant's $k$ tour.

$$\tau_{ij} \leftarrow \tau + \Delta\tau^k$$

$$\Delta\tau_{ij}^k = \begin{cases} \frac{PQ}{W} & \text{if } (i,j) \text{ is in the ant's path} \\ 0 & \text{otherwise} \end{cases}$$

Where $P = \sum_{i=1}^{N} v_i$ and $W = \sum_{i=1}^{N} w_i$ for each ant's constructed tour $N$.

### 2.4 Transition Rule

Ants select the next node based on the stochastic, fitness proportional selection. The choice to travel to node $j$ from the ant's current node $i$ is biased by the "fitness" of that edge, determined by the edge's pheromone levels and its heuristic. This creates a bias for exploration during the initial stages of the search, but after more pheromone is deposited, a bias towards exploitation of better routes. The chance to always select a sub-optimal node also helps the algorithm escape sub-optimal paths.

During the ants traversal of the graph, at any one point the ant can travel from one node to any other in the graph. There is no "distance" constrains as in other problems, such as TSP, only the weight constraint. Suggested by M.Dorigo, only nodes where their addition to the tour would not violate the weight constraint are considered [5]. Previously visited nodes are also ignored.
The probability of ant $k$ choosing edge $ij$ at step $t$ is calculated as

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta) ij]^\beta}{\sum_{h \in J_i^k} [\tau_{ih}(t)]^\alpha \cdot [\eta_{ih}(t)]^\beta} & \text{if } j \text{ is valid} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

$\alpha$ and $\beta$ are two fixed parameters that greatly affect the search.$\alpha$ represents the relative importance of the edge and $\beta$ the relative importance, creating a trade-off between the two and thus a greedy constructive heuristic.

### 2.5 Code Implementation

The algorithm was implemented in Rust, a compiled language, for its high performance time, type system, and safety guarantees. Several optimizations were made throughout the implementation.

- Nodes were stored as structs and instantiated inside dynamic arrays. This allowed for fixed indexing to an iterable data-structure while not risking stack overflow errors on large datasets, a common error with arrays. Only a pointer sized index to this data, or reference too said index, was passed to functions, rather then the object itself, greatly improving performance.

- Pheromones were stored inside a matrix with edge validation to ensure bi-directionality.

- For every node, $\eta$ and $\eta^\beta$ was pre-calculated before the search and stored within the node for efficiency.
- Extremely small $\tau$ values were not evaporated, as when close to 0, their affect is minimal but their calculation is costly to perform and risks overflow errors.

## 3 Results

### 3.1 Experiment Methodology

The experiment conducted investigated the affects of 3 parameters, population size, pheromone rate and evaporation rate. Since no parameter can be tested without the others also involved, a set of default parameters were derived through preliminary experiments, so that each parameter could be tested independently of the other changes. For each parameter tested, the others would stick to these default levels.
Each tested parameter involved a set of 8 different parameter settings. The algorithm would be run at each of these levels 20 times, to gather a range of results, each run consisting of 10,000 fitness evaluations.
The algorithm used the provided problem set of 100 nodes and a maximum weight of 295.
The default parameters are shown in table 1 and the values used for tested parameters are shown in table 2, where $p$ is the ant population size, $m$ is the pheromone rate and $e$ is the evaporation rate.

Table 1: Tested values of given parameters

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **p** | 2 | 5 | 10 | 15 | 20 | 30 | 50 | 100 |
| **m** | 0.5 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| **e** | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.8 |

### 3.2 Findings

The population of ants appeared to have the largest effect when compared to other parameters. At lower populations of $p < 20$, the algorithm performs worse than all the parameter settings. However, it rapidly grows to achieve the best performance at $p = 100$. Seen in figure 2, the lowest population of ants does not result in a better score than the initial search. However, at higher populations the algorithm beats the initial search by a large margin.
Changing the pheromone rate had very little effect regardless of the level, consistently beating the initial search at an almost constant rate.
Evaporation rate had a significant affect on the quality of found solutions, were its growth was inversely proportional to the value of solutions. A very small evaporation rate produced the highest value solutions.

Table 2: Default Parameters

| **Parameter** | $\alpha$ | $\beta$ | $e$ | $m$ | $p$ | Evals |
|---|---|---|---|---|---|---|
| **Value** | 1 | 2 | 0.5 | 1 | 30 | 10,000 |

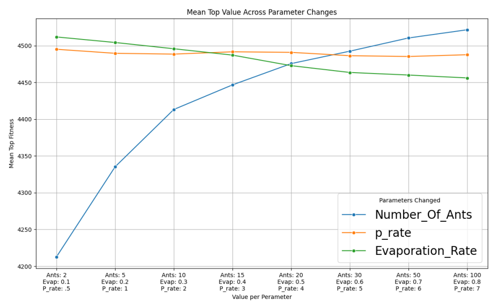

Figure 1: Performance of each parameter change

## 4 Discussion and Further Work

### 4.1 Q1: Which Combination of parameters produces the best results?

By selecting the top parameters from the experiment, of values $p = 100$, $m = 1$ $e = 0.1$, the algorithm failed to produce a single top-path where the total value was less than 4500. Even when adjusting ant population down to 50, the same performance was measured. These parameters could also find these top solutions in 1000 fitness evaluations rather than 10,000.

### 4.2 Q2/3: What do you think is the reason for the findings in the above section and what influence do the parameters have on the performance of the algorithm?

A high population allows for a larger exploration of the search space and a low evaporation rate allows for suboptimal edges to survive for longer. With less ants, this would punish the performance and either not produce high value tours at all or allow for sub-optimal paths to be encouraged early on due to the lack of exploration by extra ants.
However, due to the higher number of ants, there's

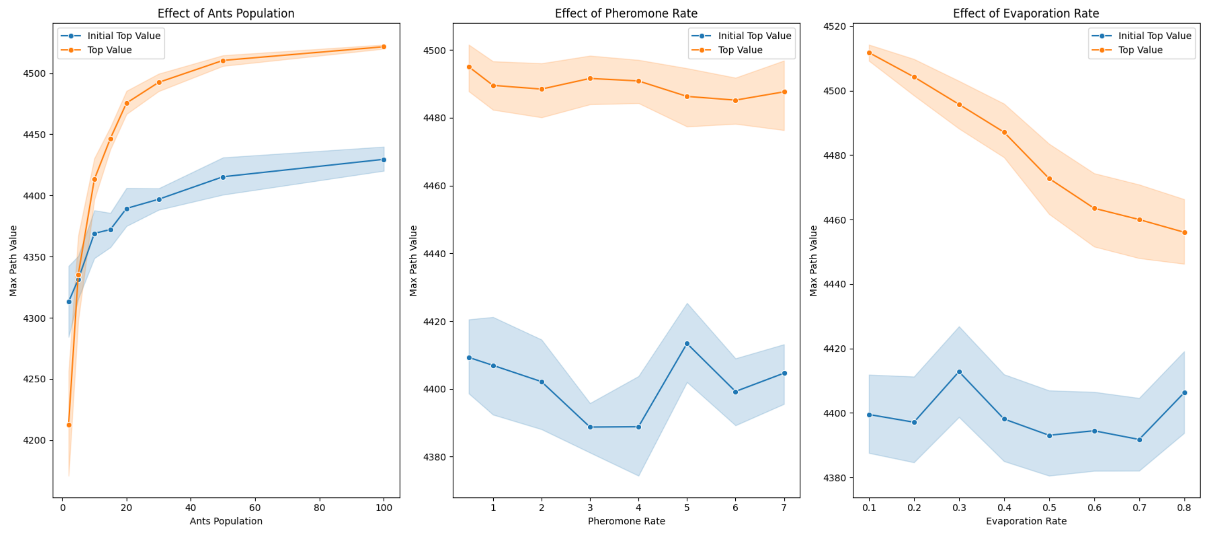

Figure 2: Difference ACO's Final Best Path vs first iteration's initial Path, per parameter change.

enough selections per iteration that a smaller probability of selecting better paths isn't punished to the same extent, as they will still be chosen by some number of ants, thus resulting in higher value edges still being exploited without the cost to exploration.

A higher evaporation rate leads to a higher convergence of solutions due to the greater punishment of worse solutions; however, this convergence is not necessarily towards optimal solutions and might lead to premature convergence.
A higher population increases the extent of the search and leads to finding better solutions but greatly affects the runtime performance of the algorithm. According to the data, the effect of higher population is logarithmic. As seen, there is relatively little difference between 50 & 100 ants compared to 5 & 50 ants. Because of this, 50 seems to be the optimal population when balancing between runtime performance and finding optimal solution.

Pheromone rate seems to have little influence as its applied equally to all tours of ants, so even better or worse tours will increase the pheromone level by the same amount each iteration. A more impactful parameter to examine would have been the $\alpha$ value, which increase the bias towards the level of $\tau$ on edges, see equation 2.

### 4.3 Q4: Might have another algorithm from the literature review provided better results?

MMAS might have provided higher value solutions in less iterations. The algorithm balances between exploration and exploitation of tours better than this paper's due to its constriction of $\tau$ levels, but also implements a form of self adaption through its frequent updating of this $\tau_{\min}$ and $\tau_{\max}$ according to the *best-so-far* tour. MMAS has also been shown to outperform similar ACO algorithms, like the one implemented in this paper, when applied to knapsack problems and other NP-Hard problems[19].

### 4.4 Further Work

To gain a better understanding of the implementation's performance and the impact of the tested parameters, the algorithm needs to be tested on larger problem sets $N > 1000$ [15]. It's possible that large population of ants visit the majority of the best nodes in the first several searches by pure chance. Since at a population of 100 and a graph of size $N = 100$, there's a strong chance all nodes might be visited by at-least one ant. $p = 100, \quad p = N$. This could produce misleading results on smaller problem sets by suggesting the algorithm is strong than it is. However, when the algorithm was ran with $p = 50, N = 500$ the same patterns were observed, with a greater difference between the initial search and the final search, suggesting the algorithm is effective at finding higher-value solutions.

## References

[1] E Aarts and amp; Laarhoven. *Simulated annealing Citation for published version (APA)*. 1997.

[2] Abdullah Alzaqebah and Ahmad Adel Abu-Shareha. Ant colony system algorithm with dynamic pheromone updating for 0/1 knapsack problem. *International Journal of Intelligent Systems and Applications*, 11(2):9, 2019.

[3] Paul A Beck. Annealing of cold worked metals. *Advances In Physics*, 3(11):245–324, Jul 1954.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, Apr 2002.

[5] Marco Dorigo and StutzleThomas. *Ant colony optimization / Ant colony optimization*. Mit Press, Cambridge, Mass., 2004.

[6] Bronson Duhart, Fernando Camarena, José carlos Ortíz-Bayliss, Iván Amaya, and Hugo Terashima-Marín. An experimental study on ant colony optimization hyper-heuristics for solving the knapsack problem. In *Mexican Conference on Pattern Recognition*, 2018.

[7] Claudia D'Ambrosio, Fabio Furini, Michele Monaci, and Emiliano Traversi. On the product knapsack problem. *Optimization Letters*, 12(4):691–712, Jan 2018.

[8] Nyoman Gunantara. A review of multi-objective optimization: Methods and its applications. *Cogent Engineering*, 5(1), Jul 2018.

[9] Hossein Hemmatian, Abdolhossein Fereidoon, Ali Sadollah, and Ardeshir Bahreininejad. Optimization of laminate stacking sequence for minimizing weight and cost using elitist ant system optimization. *Advances in Engineering Software*, 57:8–18, Mar 2013.

[10] Ralph L Holloway. The casts of fossil hominid brains. *Scientific American*, 231(1):106–115, 1974.

[11] Abdullah Konak, David W. Coit, and Alice E. Smith. Multi-objective optimization using genetic algorithms: A tutorial. *Reliability Engineering System Safety*, 91(9):992–1007, Sep 2006.

[12] Rajeev Kumar and Nilanjan Banerjee. Analysis of a multiobjective evolutionary algorithm on the 0–1 knapsack problem. *Theoretical Computer Science*, 358(1):104–120, Jul 2006.

[13] Marco Laumanns, Lothar Thiele, and Eckart Zitzler. Running time analysis of evolutionary algorithms on a simplified multiobjective knapsack problem. *Natural Computing*, 3(1):37–51, 2004.

[14] Nima Moradi, Vahid Kayvanfar, and Majid Rafiee. An efficient population-based simulated annealing algorithm for 0–1 knapsack problem. *Engineering with Computers*, Jan 2021.

[15] David Pisinger. Where are the hard knapsack problems? *Computers Operations Research*, 32(9):2271–2284, Sep 2005.

[16] R.A. Rutenbar. Simulated annealing algorithms: an overview. *IEEE Circuits and Devices Magazine*, 5(1):19–26, Jan 1989.

[17] KRZYSZTOF SCHIFF, 2013.

[18] M. Srinivas and L.M. Patnaik. Genetic algorithms: a survey. *Computer*, 27(6):17–26, Jun 1994.

[19] Valeria, Fernando S Osorio, Claudio, Fernando, and Colin G Johnson. Exploratory path planning using the max-min ant system algorithm. *Kent Academic Repository (University of Kent)*, page 4229–4235, Jul 2016.

[20] Yuan Yuan, Hua Xu, and Bo Wang. An improved nsga-iii procedure for evolutionary many-objective optimization. *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, Jul 2014.

[21] Q. Zhang and H. Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, Dec 2007.