Revisiting Multi-type Ant Colony: The Edge Disjoint Paths Problem

Robin De Haes, Mudabbir Faheem Hamza, Dieter Vandesande and Nigel Vinckier

Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium
(firstname.middlename.lastname@vub.be)

Abstract

In their paper Multi-type Ant Colony: The Edge Disjoint Paths Problem (2004) Vrancx et al. proposed a variation of the Ant Colony System (Dorigo and Gambardella, 1997) that allows finding disjoint paths in a graph. In this paper we revisit their algorithm and experiments. We try to reproduce the original results and extend their experiments in areas that were left unexplored in the original paper. More specifically, preliminary results seemed to indicate optimal solutions are generally found quickly or not at all. To mitigate this, we propose to restart the search when no progress is made. Since some uncertainties were encountered during the reproduction, this paper also addresses multiple small variations of the algorithm of Vrancx et al. as we were unable to completely reconstruct it based on the given descriptions. However, the main idea and conclusions of the original paper remain the same and are eventually reconfirmed.

1 Introduction

In nature, social insects, such as ants, work together as a colony to achieve tasks of high complexity that they could not perform on their own. However, this collaborative behavior and its results emerge without any single ant explicitly coordinating the actions of others. Instead they achieve this swarm intelligence by distributed communication via *stigmergy*, which is defined as a mechanism in which agents communicate by leaving a trace in an environmental medium to influence actions of others instead of using direct contact (Heylighen, 2016).

This principle of stigmergy and social insects were the inspiration for Ant System and many follow-up ant colony optimization algorithms (Dorigo et al., 1999). Individual agents, which we will call ants from this point on, try to solve a problem by modifying their environment which causes other agents to notice and respond to these modifications.

In the Multi-type Ant Colony System (Vrancx et al., 2004), as well as in its single-type predecessor Ant Colony System (ACS) (Dorigo and Gambardella, 1997), a problem is depicted as a graph and its solution is one or more optimal paths across this graph. These paths are searched for via a stigmergy implementation based on pheromone trails left on edges, with each ant having a certain type of pheromone. Ants will be attracted to regions in the environment that have pheromones of the same type as theirs, while also being repulsed by any other type of pheromones. This system of collaboration between fellow ants and competition with other ants was used by Vrancx et al. to find paths that are edge disjoint.

Since the time of publication of the original paper (Vrancx et al., 2004), the idea of multi-type ant colonies has inspired some further extensions that were applied to interesting use cases. These applications range from more classical problems such as preventing single points of failure in optical networks (Vrancx et al., 2006) to rather novel usages such as the study of the socio-cognitive phenomenon of perspective taking (Sekara et al., 2015).

In section 2, we will briefly introduce the progenitorial single-type ACS, as it will facilitate understanding the multi-type extension. The multi-type approach and the search heuristics that were presented in the original paper are explained in section 3. Since we were uncertain about the exact implementation of the algorithm, we will also discuss some variations we tried out in section 4 and mention which variation we used to eventually conduct the experiments. In section 5, we first briefly explain the original experiments and how we tried to reproduce them. Finally, we will end that section by presenting our own experiment investigating the effect of extending the system with search restarts. Results of the experiments are intertwined with the explanations. We will conclude in section 6 with a short discussion and summary of the findings.

2 Single-type Ant Colony System

The Ant Colony System of Dorigo (1997) consists of simple agents, i.e., ants, that cooperate to find good solutions for optimization problems such as the Traveling Salesman Problem (TSP). In TSP a shortest tour has to be found in a graph so each node is visited only once. Just like real ants, the ant agents will communicate indirectly by laying and following pheromone trails while they are searching for good tours between a source and a destination. Ants guide other

ants towards interesting regions in the search space by leaving pheromones on interesting edges. Other ants will then be attracted to the edges with more pheromones.

In practice, ant k exploits pheromone information by choosing an unvisited neighboring node j from a list of possible candidates J_k^r using the following stochastic greedy rule:

$$j = \begin{cases} argMax_{u \in J_k^r} \left[\tau(r,u)\right] \cdot \left[\eta(r,u)\right]^\beta & \text{if } q \leq q_0 \\ J & \text{if } q > q_0 \end{cases} \quad (1)$$

In this formula $\tau(r,u)$ is the amount of pheromone on edge (r,u) and $\eta(r,u)$ is a heuristic rating, which is the inverse of the edge length in our case. Therefore, ants are actually guided while building their tours by both heuristic and pheromone information. Edges that are short and have a lot of pheromones are preferred. The importance of the heuristic information can be tuned with β. The value q is a random number that is uniformly distributed in the interval [0,1]. This means the parameter q_0 determines whether exploitation or exploration is more important by having respectively a higher or a lower q_0. However, even if $q > q_0$ and the ant chooses to explore, this will happen in a biased way that exploits accumulated knowledge since the next node j is chosen using the following pseudo-random-proportional rule:

$$p^k(r,u) = \frac{\left[\tau(r,u)\right] \cdot \left[\eta(r,u)\right]^\beta}{\sum_{l \in J_r^k} \left[\tau(r,l)\right] \cdot \left[\eta(r,l)\right]^\beta} \quad (2)$$

The actual updating of the pheromone trails happens via two rules. While building its tour, each ant applies a local update rule to change the amount of pheromone on the edge it traverses. A global update rule is applied after all ants have completed their tour and is meant to reward the best tour that was found with additional pheromone deposits. In case of the Traveling Salesman Problem the best tour is simply the shortest one, which means the objective function we want to minimize is the inverse of path length.

The local update rule is given by the formula:

$$\tau(r,u) = (1-\rho)\tau(r,u) + \rho\tau_0 \quad (3)$$

The parameter ρ simulates evaporation or pheromone decay, while the parameter τ_0 represents the initial amount of pheromone present on all edges. Local updates cause the pheromone intensity to evolve back towards this τ_0. This rule is mostly meant to undo some of the pheromone changes of the global update to prevent early convergence to one path.

The global update rule is given by the formula:

$$\tau(r,u) = (1-\rho)\tau(r,u) + \rho(1/L^+) \quad (4)$$

This global update rule is only applied on edges belonging to the best solution so far, as that's the only one that should be rewarded. After this global update, a next iteration begins where we drop all ants on the initial node again and let them search for a new tour in the updated environment. This process is repeated until a good enough solution or a maximal number of iterations is reached.

3 Multi-type Ant Colony System

In single-type ACS all ants cooperate since they all have the same type of pheromone. In multi-type ACS competition is introduced on top of the cooperation by having ants that deposit different types of pheromone. Ants of the same type cooperate by depositing and following each other's pheromone trails, as was the case for the single-type system. However, ants of different types compete with each other for the best path as each type's pheromones repel other ant types. This competition is intended to make different ant types find disjoint solutions, while still having cooperation for the same ant type in order to ensure each disjoint solution is also a good path on its own. In addition to keeping track of multiple ant pheromones on each edge, two important extensions had to be made. The ant's decision process should involve a repellent factor and the objective function should take edge disjointness into account.

Extended Decision Process

The stochastic greedy rule that ants use to choose their next node gets an additional factor $\varphi(r,u)$ representing the amount of foreign pheromones on the edge between nodes r and u. This factor is the sum of all pheromones left by ants of another type. This transforms formula (1) and (2) into the following formulas respectively:

$$\begin{aligned} &\text{if } q \leq q_0\text{:} \\ &\quad j = argMax_{u \in J_k^r}\left[\tau(r,u)\right] \cdot \left[\eta(r,u)\right]^\beta \cdot \left[1/\varphi(r,u)\right]^\gamma \\ &\text{if } q > q_0\text{:} \\ &\quad j = J, \end{aligned} \quad (5)$$

$$p^k(r,u) = \frac{\left[\tau(r,u)\right] \cdot \left[\eta(r,u)\right]^\beta \cdot \left[1/\varphi(r,u)\right]^\gamma}{\sum_{l \in J_r^k}\left[\tau(r,l)\right] \cdot \left[\eta(r,l)\right]^\beta \cdot \left[1/\varphi(r,l)\right]^\gamma} \quad (6)$$

The fact that the factor φ is used in an inverted way in the formula ensures that the probability of choosing an edge decreases if a lot of foreign pheromones are present. The relative importance of the presence of foreign pheromones in the ant's decision can be tuned via the parameter γ.

Extended Objective Function

Since the main goal when solving the edge disjoint paths problem is edge disjointness, ants don't just search for the shortest path possible but primarily try to minimize sharing

edges with competing ant types. This is represented in the best path for an ant type being the one that minimizes the following objective function for its path P:

$$\sum_{e \in sharedEdges(P)} cost(e) \cdot fa(e) \tag{7}$$

In this formula, $fa(e)$ indicates how many foreign ant types use edge e, while $cost(e)$ represents the edge weight or length. Having any combination of paths in which there are no shared edges will make this sum 0, regardless of the length of the paths. If no completely disjoint solution is possible, the best solution is the one that has a minimal number of shared edges and paths that are as short as possible. When two paths have the same value for equation (7), the one with the smallest length is preferred.

Vrancx et al. (2004) also proposed an alternative objective function for situations where full edge disjointness is not possible. If a more distributed use of resources is the goal, we might prefer ants to maximally distribute themselves among the edges that are shared independent of the path length. The objective function proposed with this goal in mind is:

$$\frac{1}{||sharedEdges(P))||} \sum_{e \in sharedEdges(P)} fa(e) \tag{8}$$

Minimizing this objective function disregards the length of path P and the number of shared edges. Its main goal is obtaining an equally distributed solution.

4 Algorithm Interpretation

Our implementation of the algorithm is mostly based on the pseudocode provided by Vrancx et al. (2004). However, the pseudocode seems to diverge from the traditional implementation of single-type ACS at some unexpected points. This led us to experimenting with multiple variations. Finally, it seemed most appropriate to closely follow the pseudocode provided by Vrancx et al. Therefore, we only deviated from it if it really seemed to make more sense to do so.

In this section we will give a brief overview of why some parts of the algorithm gave rise to doubts and what the potential impact of the assumptions we made could be. We will also specify which interpretations we eventually used when reproducing the experiments. Other implementations can be found in the GitHub repository mentioned in the appendix.

Local Pheromone Updating

In the single-type Ant Colony System, Dorigo and Gambardella (1997) mentioned two possible local updating approaches. In the basic version all ants build their tours in parallel, i.e., they take one step at the same time and then all perform the local update. This way the first ant that takes a step does not influence the second ant's step by already having deposited its pheromones. The second possibility is a more sequential approach, in which each ant completes an entire tour before the next ant starts. In this approach the local updates of the first ant already influence the decisions of later ants in the same iteration. Since local updating decreases the amount of pheromones on an edge each time an ant traverses it, a more sequential search regime would lead to early ants searching closer to the currently best tour while later ants would search in a broader neighborhood.

The pseudocode provided in the multi-type Ant Colony paper (Vrancx et al., 2004) seems to make use of a more sequential implementation, as indicated by the use of many nested loops. However, it is equally plausible that some for-loops should actually be interpreted as an action that needs to be executed for multiple agents in parallel. We implemented both versions and tried to reproduce the first experiment. Although both implementations gave satisfying results, neither of them had the same results as Vrancx et al. (2004). Finally, we decided to use the sequential local updating approach since it seemed to be the most straightforward interpretation. The parallel approach would mean some for-loops should be interpreted as parallel execution, while others should still be sequential even though both type of loops are represented the same.

Global Pheromone Updating

When comparing the global pheromone updating rule mentioned in the pseudocode, we noticed it differed from the global updating rule that was presented in Dorigo and Gambardella's paper (1997). Since this difference was not mentioned by Vrancx et al. in their paper (2004), we are unsure whether the differences were intentional or not.

Global Pheromone Deposits The global update rule of formula (4) deposits an additional amount of $\rho(1/L^+)$, while the global update rule mentioned in pseudocode by Vrancx et al. (2004) deposits an additional amount of $1/L^+$. Since the experiments use a fixed value of 0.1 for ρ, this difference can have a big impact. After experimenting with both variants, we decided to deviate from the pseudocode and let the global update rule use a deposit of $\rho(1/L^+)$ instead. The large amount of pheromones that is deposited would otherwise lead to very strong attraction to the first best paths that were found. This seemed to strongly hinder exploration.

Global Pheromone Evaporation Dorigo and Gambardella (1997) also mention two variants for the global updating rule. In the first variant, the global updating rule simply reinforces the edges on the best tour. In the second variant, the global update does not only reinforce the edges on the best tour but also causes additional evaporation of pheromones on all other edges.

Based on the descriptions and pseudocode given by Vrancx et al. (2004), it seems more plausible that global

pheromone evaporation is not used. Furthermore, we experimented with both variants. Since global pheromone evaporation did not give better results, we decided to not include global pheromone evaporation.

Shared Edges Objective Function In the pseudocode of Vrancx et al. the global update rule computes the length of each path and keeps the best length found so far. It was unclear how this pseudocode relates to the shared edges objective function that was explained in formula (7). Therefore, we tried out multiple variations with varying success. Two of them will be briefly explained here, but other variations can be found in the GitHub repository mentioned in the appendix.

In a variation that was heavily inspired by the given pseudocode each ant type greedily selects its path based on path length. Afterwards, the shared edges sum is computed on the selected paths of all ant types and compared to the shared edges sum of the best solution so far. This variation gave very bad results for the first experiment. This is not completely unexpected as there is a high probability of the shortest paths not being the disjoint ones in practice. This variation was therefore not considered further.

In another variation we based ourselves more on general descriptions that Vrancx et al. (2004) gave throughout their paper. In this variation, we let each ant type perform one preliminary search without pheromone updating and use the found combination of paths as our baseline best solution so far. At the end of each iteration the shared edges sum is then computed for each ant type with regards to the paths in the best solution from the previous iterations. This approach makes sense as ants generally explore around their best path so far due to pheromone deposits. Comparing a new path for an ant type with previously found best paths for other ant types therefore seems to be a reliable way to eventually achieve disjoint paths. This variation gave good results and is the one we eventually used in our experiments.

5 Experimental Results

Vrancx et al. (2004) performed 3 experiments to investigate the behavior of their multi-type extension. In their first experiment they investigated the influence of the newly introduced parameters γ and q_0 on the speed of reaching optimal solutions in a fairly simple graph. Their second experiment examined the behavior of the system when there are more ant types than possible disjoint paths. In their third and final experiment they investigated the system's behavior when applied to a more complex graph with multiple ant types where each ant type could potentially find a disjoint path.

Based on the results of the reproduction of the original experiments, we added a fourth experiment with an extended version of the algorithm that uses "restarts" when improvements seems to stagnate. A detailed explanation can be found in the subsection of experiment 4.

In all experiments the following parameters were kept fixed: $\beta = 2$, $\tau = 0.05$, $\rho = 0.1$ and we use a maximum list of 5 candidate nodes to choose from at each step. These values are based on Dorigo's work on single-type ACS (1999) and seemed to be robust.

Experiment 1: Finding Disjoint Solutions

In this experiment Vrancx et al. (2004) used their algorithm on the simple graph of Fig. 1 with value combinations of γ and q_0 in the interval [0, 0.9] and [0, 5] respectively. The main goal is to see what the effect of these parameters are on finding completely disjoint solutions. There is only one solution that allows disjoint paths, which is the combination of path 0-1-3 and 0-2-3. Neither path is optimal on its own, since they have a path weight of 4 each and path 0-1-2-3 only has a path weight of 3. However, it is the only combined solution that is edge disjoint so for multi-type ACS this solution is the optimal one. Since the test graph is simple, only 20 iterations are given to let the algorithm converge. Finally, the average success rate of optimal convergence is computed. The results are shown in Table 1.

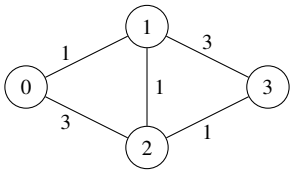

Figure 1: Test graph 1: the graph used in experiment 1. Paths will be searched from node 0 to node 3.

$q_0 \downarrow \mid \gamma \rightarrow$	0	1	2	3	4	5
0	90	100	100	100	100	100
0.1	97	98	99	99	100	100
0.2	87	93	99	99	100	100
0.3	86	95	99	99	100	99
0.4	85	99	93	98	98	99
0.5	67	90	96	97	95	97
0.6	56	80	92	93	95	99
0.7	47	70	80	90	92	95
0.8	28	48	52	73	88	92
0.9	4	20	32	44	77	84

Table 1: Percentage of optimal solutions found in test graph 1 for different values of γ and q_0. Results are averaged over 100 algorithm runs using 20 iterations and 5 ants per ant type.

Our results show that q_0 seems to have the biggest influence. Lower q_0, i.e. more exploration, leads to more alternative paths being tried out which increases the probability of finding disjoint ones. We also see γ does have an influence since a γ of 0 generally leads to worse results,

especially when combined with low exploration rates. This makes sense as the different ant types mostly try to optimize their own paths if γ is low, which guides them to the path 0-1-2-3. However, combined with a lot of exploration, the optimal solution is still frequently found.

The importance of a low q_0 is also the main conclusion Vrancx et al. made in their experiment However, it should be noted that our results differ somewhat from the original results. Although q_0 has a similar effect, increasing γ increases the success rate in our case while the performance deteriorates for γ above 2 in the original paper. Presumably, this is due to the assumptions we made in section 4.

Furthermore, the pheromone evolution in our implementation also seems to be different from the original one. To provide further insight into the inner workings of their algorithm, Vrancx et al. plotted the pheromone intensity on the different paths during a run. It should be noted that the exact value of γ and q_0 was not specified by Vrancx et al. Moreover, their plot shows 25 iterations while experiment 1 actually only has 20 iterations. Nevertheless, we tried to make a similar plot by performing a run of 25 iterations with a γ of 1 and q_0 of 0.1. The results of this run are shown in Fig. 2.

We would expect both ant types to have an increase on the disjoint paths and a decrease on the other ones as the run progresses. This evolution is noticeable in both our plot and the one made by the original authors, even though the actual pheromone intensity is quite different. The plot of Vrancx et al. has the initial amount of pheromone τ_0 as an upper bound, while it serves as a lower bound in our case. However, the local update rule of formula (3) will always make the amount of pheromone evolve towards τ_0. Since the global update only increases the amount of pheromones on the best path and does not decrease the pheromones on other paths, we would expect a plot in which τ_0 serves as a lower bound and not as an upper bound.

A possible explanation would be that Vrancx et al. used an ACS variant with global evaporation, i.e., all edges that don't belong to the best path lose some pheromones during the global update as well. They do, however, describe the single-type ACS variant without global evaporation as basis for their extension. Furthermore, their description of the plot doesn't explain how pheromone intensity for a path is actually computed. We assumed and computed the pheromone intensity averaged over all edges in a path, but Vrancx et al. could have done something else instead.

Nevertheless, the tables and plots seem to indicate that our implementation converges a bit more quickly to the disjoint paths than the one of Vrancx et al. A difference in pheromone evolution could play a role in this phenomenon.

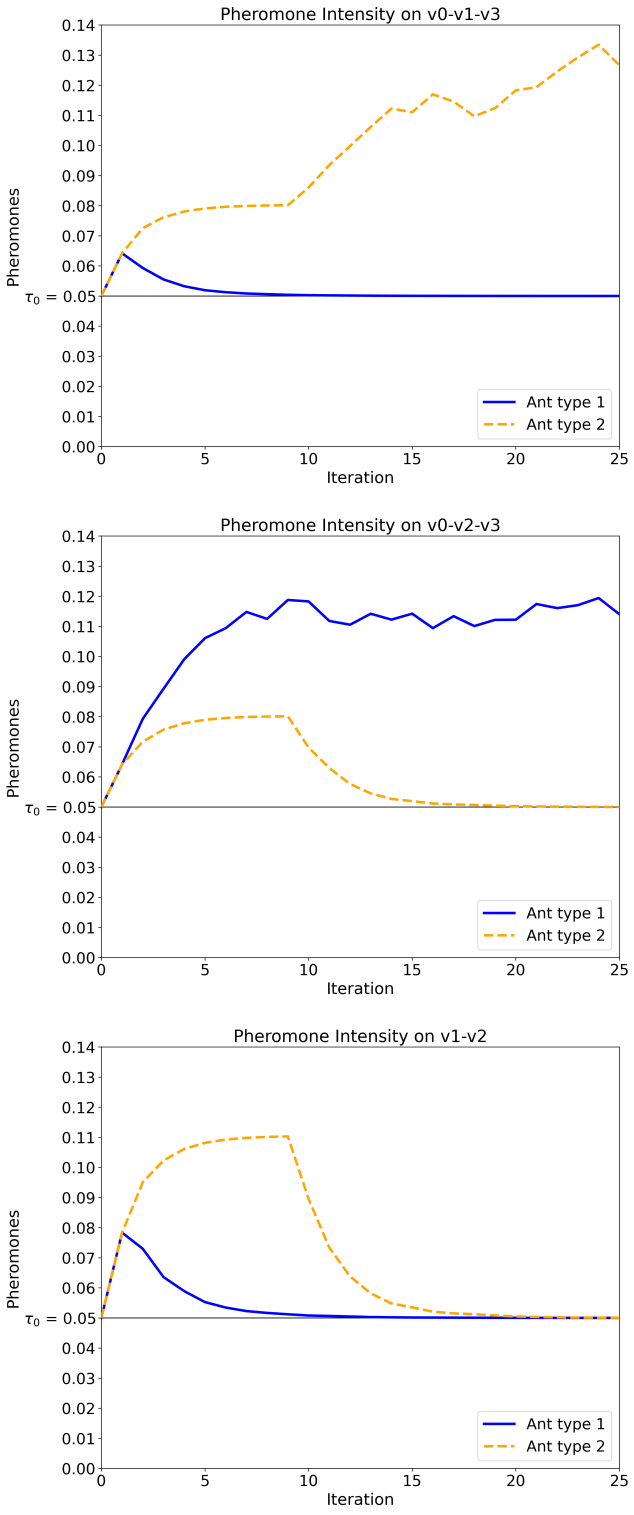

Figure 2: Evolution of pheromones on the different paths in test graph 1 during experiment 1.

Experiment 2: Sharing Paths

In this experiment Vrancx et al. (2004) investigated the behavior of their algorithm when there are more ant types than available disjoint paths. They tested out their algorithm on the graph presented in Fig. 3. This graph contains two

bridges, which forces the ant types to share a bridge whenever there are more than two of them. First a setup is tested where both bridges have the same weight and then a setup where bridge 1 has three times the weight of bridge 2. For both setups we computed the percentage of ant types that choose each bridge as part of their final solution.

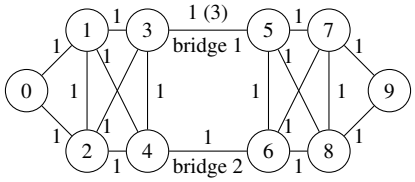

Figure 3: Test graph 2: the graph used in experiment 2. In the first part of the experiment bridge 1 has a weight of 1, while in the second part bridge 1 has a weight of 3. Paths will be searched from node 0 to node 9.

The main goal of this experiment is demonstrating the influence of the used objective function on the paths that are obtained. The algorithm is first run using the objective function of formula (7) and then using the function of formula (8). Our results for these objective functions can be respectively found in Table 2 and Table 3.

	equal weights		***different weights***	
types	**bridge 1**	**bridge 2**	**bridge 1**	**bridge 2**
2	50.5	49.5	50.0	50.0
3	47.0	53.0	33.33	66.67
4	50.1	49.9	24.75	75.25
5	50.8	49.2	21.2	78.8
6	50.0	50.0	28.7	71.3

Table 2: Percentage of ant types selecting each bridge in test graph 2 when using the objective function from formula (7). Results are averaged over 100 algorithm runs using 1000 iterations, $\gamma = 2$, $q_0 = 0.1$ and 12 ants per ant type.

	equal weights		***different weights***	
types	**bridge 1**	**bridge 2**	**bridge 1**	**bridge 2**
2	50.5	49.5	50.0	50.0
3	52.0	48.0	45.0	55.0
4	50.1	49.9	45.0	55.0
5	51.0	49.0	44.6	55.4
6	51.0	49.0	48.17	51.83

Table 3: Percentage of ant types selecting each bridge in test graph 2 when using the objective function from formula (8). Results are averaged over 100 algorithm runs using 1000 iterations, $\gamma = 2$, $q_0 = 0.1$ and 12 ants per ant type.

The objective function of formula (7) has 2 factors to optimize for. The first one is the number of ant types an edge is shared with and the second one is edge weight. The first factor will generally have the biggest impact as it is the only factor that allows fully minimizing the sum to 0. Therefore, in the case of an equal amount of ant types and bridges we'd expect them to distribute themselves equally over both bridges. When both bridges have equal weights, the edge weight factor has no influence either so again we'd expect the same distribution. When we have more ant types than bridges and a difference in edge weight, the edge weight will start to have an effect on the objective function's value. Therefore, we'd expect the lower weighted bridge to be selected more in such cases. The expected behavior that we described is clearly visible in results shown in Table 2. These results also closely resemble those originally found by Vrancx et al.

The alternative objective function of formula (8) does not take edge weight or number of shared edges into account. Instead it tries to minimize the average number of ant types that share an edge. Since edge weight does not play a role and the minimum number of shared edges is always two, the only factor that can be minimized is the number of ant types a bridge is shared with. Therefore, we expect the ants to always distribute themselves evenly over both bridges. The results in Table 3 seem to validate these expectations and again closely resemble those originally found by Vrancx et al.

Experiment 3: Multiple Disjoint Solutions

In this experiment Vrancx et al. (2004) examined the system's behavior in a more complex graph where theoretically a disjoint solution could be found for all ant types. In their experiment they mainly focused on the effect of the exploration parameter q_0, presumably because experiment 1 already showed that q_0 had the most influence on convergence to optimal solutions in a reasonable time.

The graph that is used for this experiment can be found in Fig. 4. This graph allows a maximum of 4 completely disjoint paths. Therefore, the experiment will show results for up to 4 ant types.

The experiment consists of two parts. In the first part, the regular implementation of the system is tested for a range of fixed values of q_0. For each q_0 and number of types, we compute the number of disjoint solutions. These solutions minimize the objective function of formula (7) to 0 and actually solve the edge disjoint paths problem. However, a more optimal solution would be a combination of disjoint paths that are also the shortest paths possible. To investigate how often such an optimal solution is achieved, we compute how many of the found disjoint solutions have the same combined length as the optimal solution. Such solutions are in effect always optimal themselves. It's important to note that the optimal percentage is computed relative to the number of disjoint solutions and not to the total number of solutions. Finally, we also compute a coefficient of variation of

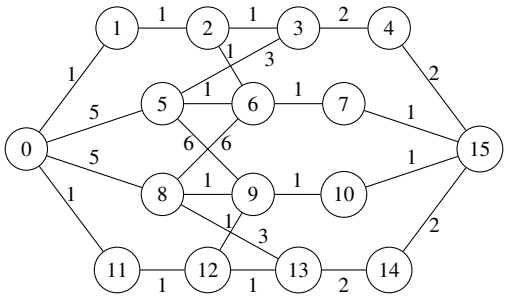

Figure 4: Test graph 3: the graph used in experiment 3. Paths will be searched from node 0 to node 15. We refer to appendix B for the optimal paths for different number of ant types.

the solutions with respect to the optimal solution. However, it should be noted that Vrancx et al. don't explain how the coefficient of variation is computed so we had to give a reasonable interpretation ourselves. Since solutions consist of paths and we want to test the variation with respect to the optimal path, computing the deviation of the total weight of all paths found in one solution with respect to the total weight of all paths in the optimal solution seemed informative. This way, our coefficient of variation indicates how much found solutions deviate from the optimal solution with regard to the total weight of their paths. A more detailed explanation on this is given in appendix A.

In the second part of the experiment we use a modified implementation of the algorithm in which we don't keep the value for q_0 fixed, but let it vary from 0.0 to 0.9 during a run. In other words, we initially let the ants explore as much as possible to then later exploit their accumulated knowledge and act more optimally. This is similar to what is often done in an ϵ-greedy approach with ϵ-decay. Since no variation scheme was given by Vrancx et al., we use a simple linear scheme in which we let q_0 vary in steps of 0.1 and distribute the number of available iterations equally between all values for q_0.

The results for both parts of the experiment are presented together in Table 4, with the last row of the table showing the results of running the modified algorithm with a variable q_0.

If there are only 2 ant types the algorithm always finds the optimal solution. This is to be expected, as the regions of the optimal paths are quite distant from each other. In this case the ant types don't need to compete much for edges. When the number of ant types further increases, we notice the algorithm has more difficulty in finding the optimal or even disjoint paths. Exploration becomes more important as the best results are generally found for fixed low values of q_0 and the variable q_0. Similar findings were made in the original experiment of Vrancx et al.

Experiment 4: The Effect of Search Restarts

Experiments 1 and 3 reveal that having enough exploration is primordial for finding disjoint and optimal solutions. Furthermore, the evolution of pheromones shown in Fig. 2 seems to indicate that pheromone intensities rise rather quickly on edges once good solutions are found. This could hinder further exploration and prevent finding better ones. Some additional preliminary research, with results added in Table 6 of the appendix, further indicated that optimal solutions are generally found early in the search or not at all.

To mitigate this behavior, we took inspiration from satisfiability (SAT) solvers (Biere et al., 2009) to implement an extension of the algorithm. It is a well-known problem that SAT solvers can get stuck in a region of the search space and therefore restarting the search, while keeping some information, sometimes increases their performance (Gomes et al., 2000).

We extended our system with restarts in the following way. During a run we keep track of the number of iterations we have without finding a better solution. Whenever a certain threshold of iterations without improvement is exceeded, the search is restarted. Such a restart consists of clearing pheromone levels and the best paths so far, so the system can start exploring again in a new search. We also keep track of the best solution over all searches. The overall best solution is the solution for which the combined shared edges sum, i.e. the shared edges sum of (7) averaged over all paths in the solution, is minimal. If this combined sum would be equal for two solutions, we consider the best one to be the solution with the lowest combined path length.

Experiment 3 was executed again, but now with our modified algorithm using a restarting threshold of 50 iterations without improvement. The results are shown in Table 5.

These results show that using restarts increases the algorithm's performance on this graph to a point where it almost always finds the optimal solution. Restarts can lead to more efficient use of iterations by abandoning searches with early convergence and exploring alternative regions more. This seems to have a good influence on the overall performance in this case.

6 Conclusion

In this paper the extension of the single-type Ant Colony System to a multi-type variant, as proposed by Vrancx et al., has been revisited. Although made assumptions caused some reproduced experiments to lead to slightly different results, their general conclusions are reconfirmed. Introducing competition on top of cooperation in an Ant Colony System seems to be a valid approach for solving the edge disjoint paths problem. In addition, experimental results of our modified version seem to indicate that restarting the search process when progress is stagnating could be a promising extension. Both efficiency and optimality of the found solution seem to improve. Research related to deciding when

	2 types			*3 types*			*4 types*		
q0	**%Disj.**	**%Opt.**	**VarCoef**	**%Disj.**	**%Opt.**	**VarCoef**	**%Disj.**	**%Opt.**	**VarCoef**
0	100	100	0	100	46	0.080	100	42	0.051
0.1	100	100	0	100	48	0.077	99	33.3	0.085
0.2	100	100	0	100	38	0.082	99	46.4	0.062
0.3	100	100	0	99	47.4	0.077	93	34.2	0.067
0.4	100	100	0	99	48.2	0.069	80	48	0.060
0.5	100	100	0	96	45.9	0.110	65	41.3	0.063
0.6	100	100	0	74	40.8	0.103	54	74.5	0.032
0.7	100	100	0	64	54	0.124	40	73.6	0.034
0.8	100	100	0	39	84.1	0.068	30	81.4	0.018
0.9	100	100	0	32	100	0	28	100	0
var	100	100	0	100	47	0.076	89	44.9	0.056

Table 4: Comparison of algorithm results of finding paths in test graph 3 for different number of ant types and q_0 values. The first 9 rows show results for fixed values of q_0 and the last row results for the modified algorithm with varying q_0.The column $\%Disj$ shows the percentage of disjoint solutions that were found. The column $\%Opt$ shows the percentage of disjoint solutions that are optimal. The $VarCoef$ column shows the coefficient of variation, computed using the formulas in appendix A. Results are averaged over 100 algorithm runs using 1000 iterations, $\gamma = 2$, $q_0 = 0.1$ and 12 ants per ant type.

	3 types		*4 types*	
q0	**%Disj.**	**%Opt.**	**%Disj.**	**%Opt.**
0	100	100	100	100
0.1	100	100	100	100
0.2	100	100	100	100
0.3	100	100	99	98.95
0.4	100	100	100	100
0.5	100	100	99	100
0.6	100	100	98	100
0.7	100	100	98	100
0.8	99	100	98	100
0.9	100	100	96	100

Table 5: Results of repeating experiment 3 with our multi-type ACS variant using restarts. The algorithm restarts when no improvement has been made within the last 50 iterations. Results are averaged over 100 algorithm runs using 1000 iterations, $\gamma = 2$, $q_0 = 0.1$ and 12 ants per ant type.

to restart could be interesting future work. Finally, it should also be noted that the performed experiments show general feasibility of the approach, but were done on fairly simple graphs. Real-world graphs are often more complex and the performance on such graphs remains to be investigated in later work.

References

Biere, A., Heule, M., and van Maaren, H. (2009). *Handbook of satisfiability*, volume 185. IOS press.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Santa Fe Institute Studies on the Sciences of Complexity. Oxford University Press, New York.

Dorigo, M., Di Caro, G., and Gambardella, L. M. (1999). Ant algorithms for discrete optimization. *Artif. Life*, 5(2):137–172.

Dorigo, M. and Gambardella, L. (1997). Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, 1(1):53–66.

Gomes, C. P., Selman, B., Crato, N., and Kautz, H. (2000). Heavy-tailed phenomena in satisfiability and constraint satisfaction problems. *Journal of automated reasoning*, 24(1):67–100.

Heylighen, F. (2016). Stigmergy as a universal coordination mechanism i: Definition and components. *Cognitive Systems Research*, 38:4–13. Special Issue of Cognitive Systems Research – Human-Human Stigmergy.

Sekara, M., Kowalski, M., Byrski, A., Indurkhya, B., Kisiel-Dorohinicki, M., Samson, D., and Lenaerts, T. (2015). Multi-pheromone ant colony optimization for socio-cognitive simulation purposes. *Procedia Computer Science*, 51:954–963. International Conference On Computational Science, ICCS 2015.

Vrancx, P., Nowé, A., and Steenhaut, K. (2006). Multi-type aco for light path protection. In Tuyls, K., Hoen, P. J., Verbeeck, K., and Sen, S., editors, *Learning and Adaption in Multi-Agent Systems*, pages 207–215, Berlin, Heidelberg. Springer Berlin Heidelberg.

Vrancx, P., Nowé, A., and Verbeeck, K. (2004). Multi-type Ant Colony: The Edge Disjoint Paths Problem. In Dorigo, M., Birattari, M., Blum, C., Gambardella, L. M., Mondada, F., and Stützle, T., editors, *ANTS 2004*, Lecture Notes in Computer Science, pages 202–213, Berlin, Heidelberg. Springer.

Appendices

The source code with our implementation of the multi-type Ant Colony System and the restart extension is made publicly available in the accompanying *GitHub repository*[1].

To allow reproducibility of results, all code for executing the experiments has been provided there as well. Python Pickle files containing the results presented in this paper are also included. Finally, some more detailed explanations and figures are provided in the appendices for more insight.

A Coefficient of Variation

Traditionally, the coefficient of variation (CV) is defined as the ratio of the standard deviation to the mean. However, we don't directly have numeric values available to compute a mean and standard deviation since solutions actually consist of paths in our case. To test the variation with respect to the optimal solution, we therefore decided to compute the coefficient of variation using the following custom formula's:

$$sol\_w = \frac{\sum_{P \in solutions}^{n} |P|}{n} \tag{9}$$

$$sol\_dev = \sqrt{\frac{\sum_{r}^{n\_runs} (sol\_w_r - sol\_w_{opt})^2}{n\_runs}} \tag{10}$$

$$CV = \frac{sol\_w}{sol\_dev} \tag{11}$$

In these formula's we consider sol_w to be total weight of a complete solution, i.e., the sum of the total path weight of each ant type's solution, with n being the number of ant types and $|P|$ being the total weight of path P. We define sol_dev as the deviation to the optimal solution with n_runs being the total amount of runs that were performed, sol_w_r being the sol_w of run r and sol_w_{opt} being the sol_w of the optimal solution.

B Optimal Paths in Test Graph 3

An optimal solution in test graph 3, according to the standard used by us, consists of a set of disjoint paths for each ant type for which the combined total edge weight is also minimal. In other words, there should be no other set of disjoint paths with a lower combined total weight.

As visible in Fig. 5, we find for 2, 3 and 4 ant types an optimal solution with a total weight of 10, 20 and 30 respectively.

C Convergence to Disjoint Solutions

In Table 6 we give an overview of the quartiles, the minimum, median and maximum number of iterations that were needed to find fully disjoint and optimal solutions when running experiment 3 for 4 types of ants. These results were computed over 100 runs, but runs without finding a disjoint or optimal solution were discarded. Quartiles, minimum, median and maximum are shown instead of the mean and standard deviation due to the presence of many outliers.

These results seem to indicate that an optimal solution is generally found in the early iterations or not at all. Disjoint solutions, however, can be found at very diverse iteration steps.

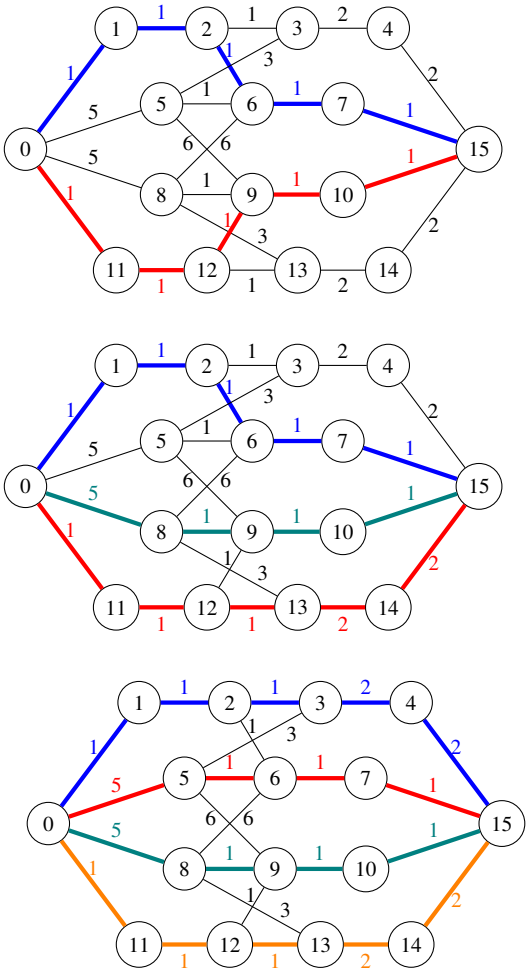

Figure 5: Test graph 3 with optimal solutions indicated for two, three and four ant types respectively. Note that for three ant types, optimal paths are not unique.

[1] https://github.com/RobinDHVUB/Multi-Type-Ant-Colony

	#It. until 1st Disjoint					*#It. until 1st Optimal*				
q0	**Min**	**Q1**	**Med**	**Q3**	**Max**	**Min**	**Q1**	**Med**	**Q3**	**Max**
0	2	10	117	176	753	2	6	9	18	50
0.1	2	9	127	188	624	2	6	8	14	37
0.2	2	55	273	431	991	2	6	10	14	28
0.3	3	13	235	343	961	3	7	11	16	30
0.4	3	17	224	343	988	3	6	9	19	86
0.5	3	13	237	444	947	3	8	14	20	41
0.6	3	8	183	349	868	3	7	9	12	30
0.7	4	9	134	38	850	4	8	14	21	48
0.8	6	12	61	46	620	6	12	20	40	67
0.9	13	20	49	54	239	13	20	28	54	239

Table 6: Overview of the quartiles, minimum, median and maximum number of iterations it took to find a disjoint or optimal solution for 4 ant types in test graph 3. The results were computed over 100 runs, with runs without finding a disjoint or optimal solution being discarded.