Ant colony optimization using two-dimensional pheromone for single-objective transport problems

Grażyna Starzec[a,*], Mateusz Starzec[a], Leszek Rutkowski[a,b], Marek Kisiel-Dorohinicki[a], Aleksander Byrski[a]

[a] *Institute of Computer Science, AGH University of Science and Technology, Al. Mickiewicza 30, Krakow, 30-059, Poland*
[b] *Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland*

Abstract

One of the acclaimed algorithms that is used to solve combinatorial graph problems is ant colony optimization (ACO). In this article, we focus on a novel extended model of the pheromone that is responsible for storing collective knowledge. The presented two-dimensional pheromone is able to accommodate more information that is extracted from feasible solutions that can be used to improve the search of a solution space. The idea is positively evaluated on TSP and VRP problems, achieving better results as compared to the original algorithm. Since it is a universal concept, it can be applied to any single-objective problem that is solvable by ACO.

Keywords: ant-colony optimisation, metaheuristics, two-dimensional pheromone

1. Introduction

In the optimization algorithms that iteratively generate feasible solutions to problems to be solved, the continued improvement of the results stems from some incorporated learning process. Most typically, an elitist approach

*Corresponding author.
Email address: gstarzec@agh.edu.pl (Grażyna Starzec)
The Version of Record is available at: https://dx.doi.org/10.1016/j.jocs.2024.102308.

is represented, which means that any information is only extracted from top solutions with the aim of directing a search toward the better regions of a solution space. In the case of the ant colony optimization algorithm (ACO), the model that is used to store the information that is extracted from the candidates is a very simple structure; the most popular variants learn only from those solutions with the lowest costs [1, 2, 3, 4].

In our previous research [5, 6], we showed that it was possible to improve the results that were obtained by ACO by using more solutions in the learning process that were based on planned-leveraging HPC to run a specific implementation of distributed ACO. Following this idea, we decided to pursue this research further and encode more information into the pheromones that were used in ACO (thus, increasing their dimensionality). In other words, each edge would contain significantly more information than before, helping in the realization of complex optimization tasks. Moreover, the algorithm should be able to make better use of computational resources and increase the diversity of a search, resulting in improved performance as compared to the reference algorithms. Finally, a possible future work is the application of the idea of a multi-dimensional pheromone for the solving of multi-objective optimization problems by ACO-type algorithms [7, 8, 9].

This article is constructed as follows. First, we discuss the background of the presented research. Then, we present the details of our proposed modification to the ant colony optimization pheromone. After this, a series of conducted experiments is described, including a discussion of their results. A summary of the article concludes the paper.

2. Research background

In order to picture the context for this research, it is worth presenting some material in two areas – the ant colony optimization algorithm, and the automatic configuration of the algorithms.

2.1. Ant Colony Optimization

The ant colony optimization algorithm was originally proposed by Marco Dorigo [10] as an algorithm for solving the traveling salesman problem (TSP) and was later presented as a metaheuristic algorithm by Dorigo and Caro [11]. It was inspired by the way that ants share their knowledge in their colonies in nature.

In its basic form, the algorithm can be used to solve single-objective optimization problems, which are represented as a graph of vertices; the edges between them have assigned costs with defined restrictions and requirements that describe those paths that can be feasible solutions to a stated problem. The solution with a minimum cost (defined as a function of the costs of its component edges) is an optimal solution.

The algorithm consists of subsequent iterations in which a population of ants (agents) generates candidate solutions. These solutions are created in a step-by-step manner and steered by a probabilistic decision rule (see Eq. 1) that determines the selection of the next vertex. This is based on two sources of information – the heuristic attractiveness, and the pheromone trail. The former is a value that indicates how promising a given edge looks when it comes to obtaining a solution with a low cost, while the latter is a value that represents the quality of those solutions that the algorithm has already found that contain this edge.

Let us introduce the variables that are present in the formula of the probabilistic decision rule. In iteration t, the probability of selecting the edge between vertices i and j by ant k is dependent on the current value of the pheromone trail of this edge $\tau_{ij}(t)$ and its heuristic attractiveness $\eta_{ij}(t)$. The relative weight of these two elements is controlled by the powers of α and β. The probability is calculated with regard to other possible moves A_k. The formula is defined as follows:

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}^\alpha(t)\,\eta_{ij}^\beta(t)}{\sum_{l \in A_k} \tau_{il}^\alpha(t)\,\eta_{il}^\beta(t)} & \text{if } j \in A_k \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

The valid solutions that have been created by the ants are used further to update the pheromone trail. The original algorithm modifies the pheromone values after each iteration, applying pheromone evaporation (controlled by $\rho \in [0, 1)$ – a pheromone-persistence coefficient) and using all m feasible solutions that have been generated in the iteration as given in the following formula:

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k. \tag{2}$$

Given L_k, the cost of a given solution, and a constant Q, the solution updates the pheromone by a value that is defined as follows:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{L_k} & \text{if } k\text{-th ant uses edge } (i,j) \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

Many modifications have been proposed to outperform the original algorithm; our area of interest has mostly focused on the pheromone and its update process. As opposed to the original version that is described above, later variants have usually proposed strongly limiting the numbers of solutions that update the pheromone. The elitist ant system (EAS, [1]) simply uses a limited number of solutions with the lowest costs. The rank-based ant system (ASRank, [2]) acts in a similar fashion as EAS, but the impact of a specific solution is weighted by its rank among the solutions so that the best solution leaves the strongest trail. Finally, the max-min ant system (MMAS, [3]) uses a mere single best solution. Incidentally, this algorithm also introduced an idea that has gained a lot of popularity: imposing pre-established bounds on the values of the pheromone trail.

Overall, the most popular variants of ACO mainly extract knowledge from the most promising solutions. Although this is a popular approach in optimization algorithms, it ignores the potential of learning from most of the solutions; these are simply ignored, causing a waste of computational effort [5].

Given the topic of our research, we should also mention a related idea of adding an additional dimension to the pheromone (described in [12]), which stores information about customer blocks and their exact locations from the VRP solutions. Even though there is a common ground among the concepts, it is important to note that the idea that is presented in [12] is strictly associated with a specific problem type being solved, whereas our goal is to propose a general problem-independent extension of the pheromone.

2.2. Automatic algorithm configuration

Metaheuristic optimization algorithms are often parameterized, which makes it difficult to objectively evaluate their performance. With exponentially growing search spaces for the multiple parameters that control the behavior of the algorithm, finding optimal values for them is a difficult hyper-optimization task *per se*. Typically, only a limited number of configurations can be tested. A simple approach to this problem is often based on selecting a few possible values of the parameters and evaluating all of the combinations. One of the publicly available software applications that is aimed at

the problem of automatic algorithm configuration in a more advanced way is *irace* [13]. This algorithm implements *iterated racing* procedures, including the iterated F-race algorithm (which, among others [14, 15, 16], have been successfully used to automatically configure multi-objective ACO algorithms [17, 18, 19]). Given an algorithm with the definitions of its parameters and the instances of a problem to be solved, the software sifts through the parameter space looking for those configurations (parameters values) that statistically perform better than the others.

3. Two-dimensional pheromone for ant colony optimization

As discussed earlier, most of the popular variants of the ACO algorithm use very limited numbers of solutions (often just a single one [3]) to update the pheromone in a single iteration. Our previous research showed, however, that this algorithm may benefit from learning from more solutions – especially when larger colonies are in use [5]. The concept that we would like to propose is based on the idea that the algorithm could be further improved by introducing a more composite model of a pheromone that could contain information from more feasible solutions.

The standard model of the pheromone is a single value of a trail that is maintained for each edge. The extended structure, in turn, could store multiple values; hence, the name "two-dimensional pheromone" (the first dimension being the index of the value, and the second – the value itself). The values would be updated independently by different groups of solutions. Such a modification requires some parts of the algorithm to be appropriately adjusted. Adapted approaches need to be proposed for the steps around the pheromone, such as its update process and its use in the ants' decision-making processes. An initial concept and a preliminary test of this idea were presented in [20].

3.1. Update process

The pheromone trail is the mechanism in the ant colony optimization algorithm that is responsible for learning from the feasible solutions that have been found thus far. In the original algorithm, this is modeled as a single value per edge in the problem graph. The values are modified by two mechanisms: evaporation, and increase; the former (also called "extinction") consists of decreasing each value by some fraction (*extinction*) after each iteration, while the latter also happens after each iteration and is responsible

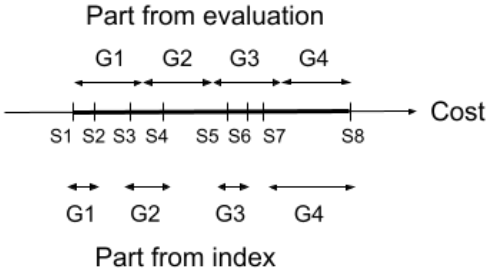

Figure 1: Grouping by part from evaluation vs. part from index

for increasing the trail by some particular value (*increment*) for those edges that were part(s) of the selected solution(s).

Our proposal introduced multiple values in lieu of a single one for each edge; this required the above mechanisms to be adjusted accordingly. The extinction mechanism was simply applied to all of the values in a similar manner as with the standard algorithm, while the other mechanism was transformed in a more complex way. Our main goal was to extract information from a larger set of solutions that was created in each iteration. Having n values of the trail for each edge, these solutions are then divided into n groups according to their cost, and each group modifies one part (a single value) of the pheromone. Two strategies were proposed for the grouping of the solutions: $PartFromIndex$ (PFI), and $PartFromEvaluation$ (PFE) (see Fig. 1).

In $PartFromIndex$, the solutions are sorted by their costs and then divided into equinumerous (or as equinumerous as possible) groups. In $PartFromEvaluation$, the range that is determined by the solution costs is divided into equal subranges, and the subrange into which a specific solution falls determines to which group it is assigned.

The update process then acts as follows:

1. For each group of solutions, the *increment* value is divided by its size in order to maintain the same amount of pheromone deposition.
2. For each edge in each solution in the group, the current value that is stored in the assigned part of this edge's pheromone is increased by the value that was calculated in the previous step.

3.2. Interpretation variants

In the original ACO algorithm, the pheromone trail that is modeled as a single value for each edge is used during the solution's construction (as presented in Formula 1). In the proposed two-dimensional model, there are multiple values instead of a single one. In order to again obtain a single value that can be used in the formula, we initially propose three variants of the interpretation of the extended pheromone structure:

- ExponentialRandom (ER) – a randomized choice of one of the available values in which the first value (i.e., the one that is updated by the best solutions) is selected in half of the cases, while each subsequent value is selected half as often as the previous one (with the exception of the last value – assuring that the probability adds up to 1).
- WeightedCombination (WC) – the values in the pheromone are combined by a weighted product with the weights assigned as follows: the first value is assigned a weight of 0.5, and the weight of each subsequent value is half the size of the weight of the previous one (again, with the exception of the last value so that all weights add up to 1).
- PairingCombination (PC) – this variant can be described as a multi-step procedure (also see Fig. 2 for reference):
  - values are paired up from outside toward center (note: in each pair, one value is updated by better solutions and another by worse solutions);
  - for each pair, average and difference are calculated;
  - difference is multiplied by index of pair (as presented in figure), which makes difference most reinforced for outer pair – average is then added to this value;
  - having calculated such values for all pairs, we finally compute their average;
  - this average is then ensured to be not larger than established $maxValue$ of pheromone and not smaller than smallest value of pheromone that is possible in given iteration (which can be calculated as $maxPhValue * (1 - extinction)^{iterationsSoFar}$).

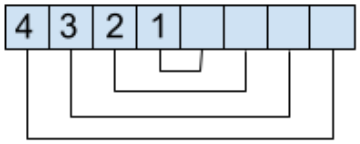

Figure 2: Pairing and indexing in PC method

These three variants were used in the first part of the experiments that are described below. For the next part of the experiments, two additional methods were proposed:

- ExponentialRandomMax (ERM) – a modification of the Exponential-Random interpretation variant where the variant can be described as follows: a random value points to the index of the pheromone from which we take the value (according to the distribution as is described for that variant); then, we select the maximum of the values that are stored in the pheromone up to the selected index in this variant.
- ExpectedCombination (EC) – this is also a multi-step procedure that has its origins in the idea of treating a set of pheromone values as a kind of probability distribution of how often a solution with this edge falls into the group that is represented by each value and calculating a sort of expected value of the edge's evaluation based on it:
  - Each value (part) of the pheromone has a corresponding score assigned to it (between 0 and 1) – assuming that the indexing of the parts starts from 0 for the part that is updated by the worst solutions, then the score of each part is $(index + 0.5)/twoDimPheromoneSize$.
  - We want to treat the values as a distribution so that they are normalized to add up to 1 (divided by their sum).
  - The expected score is calculated as a sum of the multiplications of the normalized value and assigned score.
  - The obtained value (between 0 and 1) is then adjusted to range between the minimum and maximum values of the pheromone.

4. Experimental evaluation

To evaluate the proposed modifications, we have extended the framework that was described in our previous article [20]. For the second part of the experiments, new variations of the two-dimensional pheromone were implemented, and the code was adjusted to be compatible with the *irace* software for the purpose of the automatic search of the algorithm's parameters space (including the parameters of the two-dimensional pheromone). The experiments were based on the framework's ability to solve TSP and CVRP problems, noting that no problem-specific local optimizations were applied to the created solutions that aligned with our objective to evaluate the algorithm as a problem-agnostic metaheuristic.

Some of the parameters were set to the same constant values in all of the experiments (described below):

- $iterationsNums$ — number of iterations of algorithm — 200;
- $minPhValue$ — minimum value of pheromone — 0.001;
- $maxPhValue$ — imposed maximum value of pheromone — 0.999.

4.1. Selective experiments

The first set of experiments was conducted for two larger TSP instances from TSPLIB (namely, $kroA100$ and $tsp225$) with the following values of the algorithm parameters (all possible combinations of values for the various parameters):

- $antsNum$ — number of ants in colony — 20 or 50 for $kroA100$, and 20, 50, or 100 for $tsp225$;
- α — power of pheromone in probabilistic decision rule — 2.0 or 3.0;
- β — power of heuristic value in probabilistic decision rule — 2.0 or 3.0;
- *pheromoneType* — choice between standard "one-dimensional" and experimental two-dimensional pheromones — $Basic$ or $TwoDim$;
- *pheromoneDelta* — values used for two parameters: pheromone update *increment*, and pheromone *extinction* fraction — 0.01, 0.05, or 0.1;

- *updateNum* — number of best solutions used to update pheromone — 1 (only for *Basic*), half of *antsNum*, or -1 (meaning all solutions from iteration).

For the configurations with *pheromoneType* set to $TwoDim$, a few more parameters are necessary:

- *twoDimensionalPheromoneSize* — number of values associated with single edge;
- *interpretationType* — one two-dimensional pheromone-interpretation variant —$ExponentialRandom$, $ExponentialRandomMax$, $WeightedCombination$, $PairingCombination$, or $ExpectedCombination$;
- *updateType* — one two-dimensional pheromone-update type — $PartFromEvaluation$ or $PartFromIndex$.

In order to evaluate various configurations of the algorithm, we compared them with the average global best results that were found over 20 repeats. The top-30-performing configurations are listed in Tab. 1 for *kroA*100 and Tab. 2 for *tsp*225.

Based on this, we can make the following observations: most of the configurations had the largest-possible numbers of ants and $\alpha = 2.0$, $\beta = 3.0$, *pheromoneDelta* $= 0.05$. In both tables, a configuration with a *Basic* pheromone appeared only once (in the 16th and 30th positions, respectively); this was a configuration with a single ant that updated the pheromone and the other values that were set in the same way as was described in the previous point. $TwoDim$ configurations usually use half of the population to update the pheromone. *InterpretationType* set to $ExponentialRandom$ dominated the table, while $PairedCombination$ also showed up in multiple rows, but $WeightedCombination$ appeared only once. $PairedCombination$ was usually combined with smaller $TwoDim$ pheromone sizes and the whole population that updated the pheromone. *UpdateType* set to $PartFromEvaluation$ was more popular than $PartFromIndex$. The dominance of the smaller or larger $TwoDim$ pheromone sizes was not clear (even though the first configuration used 20 in both tables).

The execution time of the algorithm was not our main concern at this point, and the code was not written in the most efficient way. Still, let us just briefly note that the configurations that used the two-dimensional pheromone took up to two-times longer than the standard algorithm with the

Table 1: Average global-best results from 20 repeats ($kroA100$)

Ants	α	β	**Pheromone**	**Inc.**	**Ext.**	**UN**	**TDS**	**InterpT.**	**UpdateT.**	**Avg. score**
50	2	3	TwoDim	0.1	0.1	25	20	ER	PFE	22168.95
50	2	3	TwoDim	0.1	0.1	25	4	ER	PFE	22238.63
50	2	3	TwoDim	0.1	0.1	25	10	ER	PFE	22272.90
50	3	3	TwoDim	0.1	0.1	25	10	ER	PFE	22273.12
20	2	3	TwoDim	0.05	0.05	10	4	PC	PFI	22311.09
50	2	2	TwoDim	0.05	0.05	-1	20	ER	PFI	22319.00
50	2	3	TwoDim	0.05	0.05	-1	20	ER	PFE	22341.16
50	2	3	TwoDim	0.05	0.05	-1	20	PC	PFI	22345.67
50	3	2	TwoDim	0.05	0.05	25	10	ER	PFE	22406.36
20	3	3	TwoDim	0.05	0.05	10	10	ER	PFE	22418.92
50	2	3	TwoDim	0.05	0.05	25	10	ER	PFE	22422.77
20	2	3	TwoDim	0.1	0.1	10	10	ER	PFE	22442.58
50	2	3	TwoDim	0.05	0.05	25	20	ER	PFE	22450.34
20	2	3	TwoDim	0.05	0.05	10	4	ER	PFE	22478.21
50	2	2	TwoDim	0.05	0.05	25	4	ER	PFE	22482.15
50	**2**	**3**	**Basic**	**0.05**	**0.05**	**1**	-	-	-	**22499.86**
50	3	3	TwoDim	0.01	0.01	-1	10	PC	PFE	22505.25
50	3	3	TwoDim	0.1	0.1	25	20	ER	PFE	22509.47
50	2	3	TwoDim	0.1	0.1	25	10	ER	PFI	22509.93
50	2	3	TwoDim	0.05	0.05	25	4	ER	PFE	22516.57
50	2	3	TwoDim	0.05	0.05	25	10	ER	PFI	22517.97
50	3	2	TwoDim	0.1	0.1	25	4	ER	PFE	22518.19
20	2	3	TwoDim	0.05	0.05	-1	4	PC	PFI	22521.77
50	3	2	TwoDim	0.05	0.05	25	20	ER	PFE	22538.59
50	2	3	TwoDim	0.05	0.05	-1	20	PC	PFE	22539.33
50	2	2	TwoDim	0.1	0.1	-1	4	PC	PFE	22543.72
50	3	3	TwoDim	0.05	0.05	-1	20	WC	PFI	22561.96
50	2	3	TwoDim	0.05	0.05	25	4	PC	PFE	22563.30
50	2	2	TwoDim	0.1	0.1	25	4	ER	PFI	22564.82
20	2	3	TwoDim	0.05	0.05	10	10	ER	PFE	22571.30

Inc. – increment; Ext. – extinction; UN – updating solution number; 2DS – two-dimensional pheromone size; InterpT. – interpretation type; UpdateT. – update type; PC – pairing combination; ER – exponential random; WC – weighted combination; PFE – part from evaluation; PFI – part from index

Table 2: Average global-best results from 20 repeats (*tsp*225)

Ants	α	β	**Pheromone**	**Inc.**	**Ext.**	**UN**	**TDS**	**InterpT.**	**UpdateT.**	**Avg. score**
50	2	3	TwoDim	0.05	0.05	25	20	ER	PFE	4116.97
100	2	3	TwoDim	0.05	0.05	50	20	ER	PFE	4127.85
100	2	2	TwoDim	0.1	0.1	-1	4	ER	PFI	4137.80
100	2	2	TwoDim	0.1	0.1	50	4	ER	PFE	4140.03
100	2	3	TwoDim	0.1	0.1	50	10	ER	PFI	4143.05
100	2	2	TwoDim	0.05	0.05	50	10	ER	PFE	4147.36
100	2	3	TwoDim	0.1	0.1	-1	4	PC	PFI	4147.72
50	2	3	TwoDim	0.05	0.05	25	4	PC	PFI	4149.90
100	2	3	TwoDim	0.05	0.05	50	10	ER	PFI	4152.27
100	2	3	TwoDim	0.05	0.05	-1	10	PC	PFI	4152.33
100	2	3	TwoDim	0.1	0.1	50	4	ER	PFE	4153.28
100	2	2	TwoDim	0.05	0.05	50	20	ER	PFI	4153.60
50	2	3	TwoDim	0.05	0.05	25	4	ER	PFE	4154.06
100	3	3	TwoDim	0.05	0.05	50	10	ER	PFE	4154.74
100	2	3	TwoDim	0.05	0.05	50	20	ER	PFI	4155.69
100	2	3	TwoDim	0.05	0.05	50	10	ER	PFE	4158.35
100	2	3	TwoDim	0.05	0.05	-1	20	ER	PFI	4158.78
100	2	3	TwoDim	0.05	0.05	-1	10	ER	PFI	4159.71
100	2	3	TwoDim	0.05	0.05	50	4	PC	PFI	4159.78
100	2	3	TwoDim	0.05	0.05	50	4	ER	PFE	4160.54
50	2	3	TwoDim	0.05	0.05	25	10	ER	PFE	4162.32
100	2	2	TwoDim	0.05	0.05	-1	10	ER	PFI	4162.72
50	2	3	TwoDim	0.05	0.05	-1	10	ER	PFI	4164.55
100	2	3	TwoDim	0.05	0.05	-1	4	PC	PFE	4164.56
100	2	3	TwoDim	0.05	0.05	-1	20	PC	PFI	4164.71
50	2	3	TwoDim	0.05	0.05	-1	10	ER	PFE	4165.18
100	2	2	TwoDim	0.1	0.1	50	10	ER	PFI	4166.39
50	3	3	TwoDim	0.05	0.05	25	20	ER	PFE	4168.79
100	2	3	TwoDim	0.1	0.1	50	10	PC	PFI	4169.98
100	**2**	**3**	**Basic**	**0.05**	**0.05**	**1**	-	-	-	**4171.44**

Inc. – increment; Ext. – extinction; UN – updating solution number; 2DS – two-dimensional pheromone size; InterpT. – interpretation type; UpdateT. – update type; PC – pairing combination; ER – exponential random; WC – weighted combination; PFE – part from evaluation; PFI – part from index

same number of ants. As we can see for both instances, however, there were configurations with the two-dimensional pheromone and a smaller number of ants that performed better than the standard algorithm (the fifth and first positions, respectively), and the execution time of these configurations was actually around 75-85% of the corresponding (and nota bene – worse performing) configurations that used the basic pheromone structure.

Overall, we can see that introducing another dimension to the pheromone and updating the pheromone by more than just a single top solution could lead to better obtained results by the ant colony optimization algorithm. Among the tested configurations, the ones that used $ExponentialRandom$ and $PartFromEvaluation$ performed especially well.

4.2. Automatic configuration for TSP

Selective experiments only allow us to test a limited number of configurations, whereas it is valuable to search the parameter space for the values that give the best results. For this, we used the *irace* [13] package with various definitions of acceptable parameters. The common parameters were configured as follows:

- iterations – constant (200);
- number of ants – constant (100);
- α – range (0.1, 5.0);
- β – range (0.1, 5.0);
- pheromoneDelta – range (0.04, 0.1) – common parameter for pheromone *increment* and *extinction* values;
- updateAnts (UN) – integer from range (1, 100) – with exception of "Single Basic" configuration (described below).

The parameters for the scenarios with the two-dimensional pheromone were configured as follows:

- twoDimPhSize – even integer from range (2, 20);
- updateType – choice of (PartFromEvaluation; PartFromIndex).

In order to test the various heterogeneous scenarios separately, we ran the following independent runs:

- Basic – with pheromoneType set to Basic;
- SingleBasic – with pheromoneType set to Basic and updateAnts set to 1;
- Five scenarios with interpretationType preset to selected value (Expected, ExponentialRandom, ExponentialRandomMax, Pairing, or Weighted);
- TwoDimAll – with pheromoneType set to TwoDim and getType configured as choice from possible values listed above.

The *irace* software was configured with its default settings with a few exceptions. Due to the high variance of the results, we decided to set the number of instances that were to be evaluated before the first elimination test to 12 instead of the default 5 (this also set the number of configurations that were sampled and evaluated at each iteration to 12); also, we set the number of instances that were evaluated between the elimination tests to 3. The budget was set to 100,000 seconds (with the exception of the TwoDimAll scenario, where it was increased to 300,000 seconds).

Since the tuning was limited by the total execution time, we selected a set of instances with fairly similar sizes. During the search phase, we used six TSP instances: kroA100, kroB150, kroC100, pr76, pr124, and pr144. The configurations that were found by *irace* for each scenario are listed in Tab. 3. These were then tested on seven other instances: kroA150, kroB100, kroD100, kroE100, pr107, pr136, and pr152. Since the result of the tuning was dependent on the training set, we wanted the test instances to be similar as well. The results that were obtained for the test instances over 20 repeats are presented as box plots in Fig. 3.

Table 3: Configurations found by *irace* for TSP

Scenario	α	β	phDelta	UN	phType	twoDimPhSize	updateT	interpretationT
Basic	1.3444	4.4230	0.0988	5	Basic	-	-	-
BasicSingle	0.8036	4.1805	0.0523	1	Basic	-	-	-
Expected	0.8264	4.8468	0.0705	6	2D	14	PFE	Expected
ExponentialRandom	0.9391	4.4132	0.0643	2	2D	2	PFE	ExpRnd
ExponentialRandomMax	0.7453	4.3232	0.0991	29	2D	20	PFE	ExpRndMax
Pairing	0.7726	4.9294	0.0929	8	2D	10	PFI	Pairing
Weighted	0.7078	4.6913	0.0580	2	2D	8	PFE	Weighted
TwoDimAll	0.9285	4.9715	0.0996	15	2D	14	PFE	ExpRnd

Looking at the charts, we can see that it is difficult to observe the strong repeatability of the better or worse performances of the configurations among the test instances. The Basic scenario resulted in a configuration with five ants updating the pheromone, but the BasicSingle mostly performed better. On most of the charts, a few of the configurations with the two-dimensional pheromone can be seen to have been better than the ones with the basic pheromone (for the pr152 problem alone, only a single one [ExpRnd] stood out as being better). In all of the scenarios, *irace* found fairly similar values of α and β. Consistent with the results from the selective experiments 4.1, the Weighted variant did not usually come out well. Furthermore, the performance of the newly proposed Expected and ExpRndMax variants was very mixed across the test instances (e.g., ExpRndMax had the best average for the kroD100 problem). Also consistent with the previous Set of Experiments 4.1 was the fact that the ExpRnd variant was usually better than the configurations with the standard version of the pheromone – both configurations from the ExpRnd and TwoDimAll scenarios. The former scenario was simplified to having 2 ants update the pheromone with a size of 2 (which was significantly different from the configurations that were tested in the selective experiments 4.1), and the latter was selected to having 15 ants update the pheromone with a size of 14 (which actually performed better – mostly). The Pairing variant presented quite good performance as well; interestingly, is was the only one that combined *irace* with PFI (similar to what was observed in 4.1). Out of the proposed variants, Pairing and ExponentialRandom generally performed the best; additionally, the second one was also selected in the TwoDimAll scenario.

Thanks to making the code more efficient, we observed a lower time overhead for the two-dimensional pheromone. For example, one of the best-performing configurations, the one that was selected in the TwoDimAll scenario, was around 15% slower than the BasicSingle configuration.

4.3. Automatic configuration for VRP

A similar experiment was conducted for VRP with *irace*. Similar to what was done for TSP, we again selected a set of instances of fairly similar sizes for both the training and testing. During the search phase, we used six TSP instances: kroA100, kroB150, kroC100, pr76, pr124, and pr144. The configurations that were found by *irace* for each scenario are listed in Tab. 3. These were then tested on seven other instances: kroA150, kroB100, kroD100, kroE100, pr107, pr136, and pr152. Since the result of the tuning

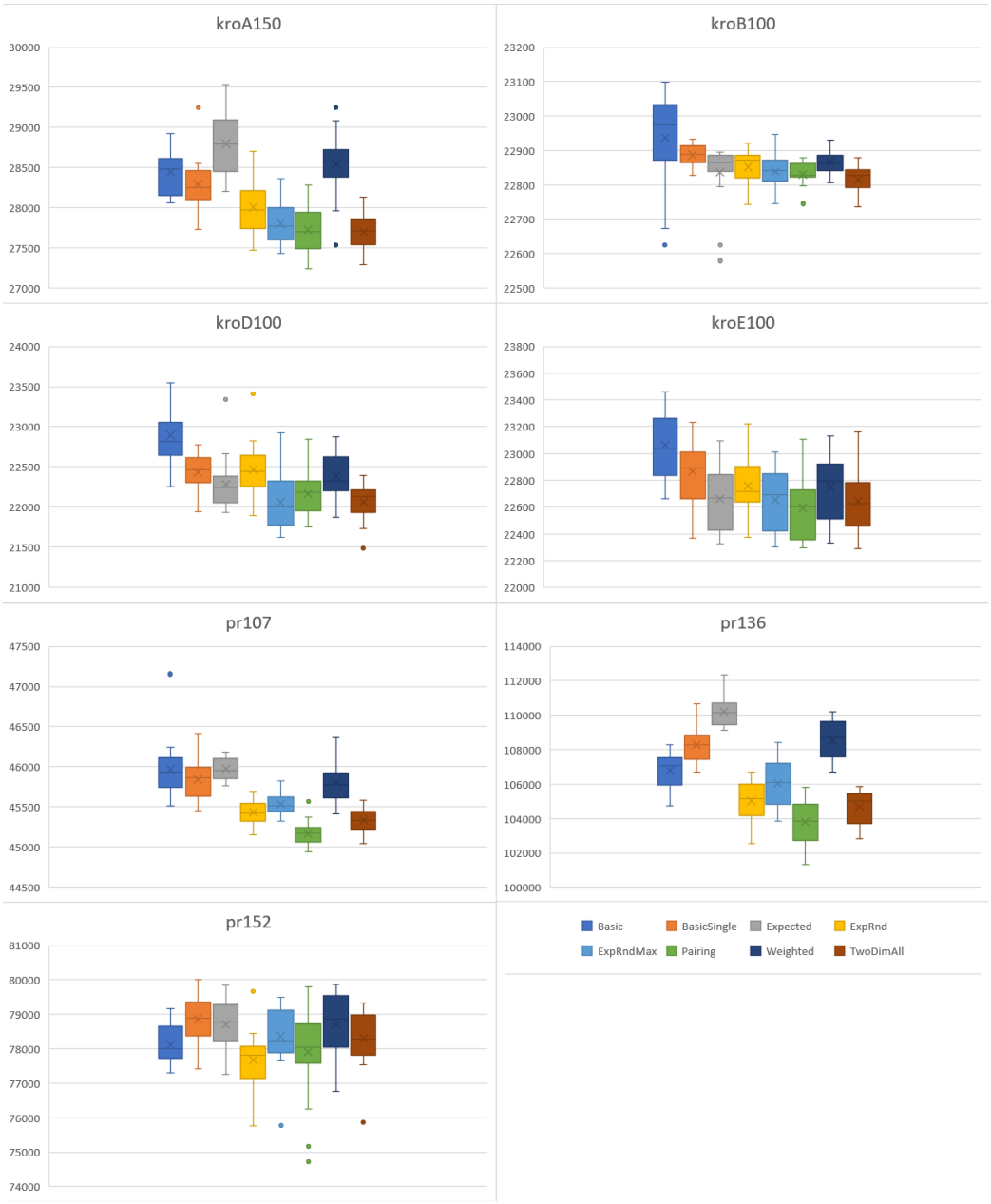

Figure 3: Results for *irace* TSP experiment

was dependent on the training set, we wanted the test instances to be similar as well. In the search phase, we used six instances: tai150d, tai150b, CMT4, M-n121-k7, CMT13, and E-n101-k8. Tab. 4 lists the configurations that were found by *irace*. These were then tested on six other instances: tai150c, tai150a, CMT9, M-n151-k12, CMT11, and P-n101-k4. Fig. 4 presents the results that were obtained by these as box plots for the test instances over 20 repeats. Since the number of vehicles in the solutions was always the same across all repeats and configurations (for every test instance), it was excluded from the analysis of the solutions cost and the charts only present the total distance.

Table 4: Configurations found by *irace* for VRP

Scenario	α	β	phDelta	UN	phType	phSize	updateT	interprationT
Basic	1.1457	2.564	0.0834	1	Basic	-	-	-
BasicSingle	0.9957	3.5957	0.0859	1	Basic	-	-	-
Expected	1.1924	3.1466	0.0804	2	2D	20	PFE	Expected
ExponentialRandom	2.2521	2.7513	0.0633	36	2D	18	PFE	ExpRnd
ExponentialRandomMax	1.0863	4.1444	0.0978	38	2D	10	PFE	ExpRndMax
Pairing	1.0858	3.6955	0.0613	57	2D	12	PFE	Pairing
Weighted	0.4803	4.7727	0.0738	30	2D	12	PFE	Weighted
TwoDimAll	1.3649	4.7456	0.0566	24	2D	8	PFE	Pairing

In general (here as well), it is difficult to mark the repeatability of the configurations' efficiency among the test instances. The Basic scenario resulted in a configuration with just a single ant updating the pheromone, but BasicSingle led to different values of primarily α and β and mostly performed better. In those scenarios with the two-dimensional pheromone, *irace* decided to choose more ants to update the pheromone than for TSP (4.2). On most of the charts, at least one of the two-dimensional configurations can be considered to be better than the ones with the basic pheromone (better results of multiple variants were obtained for the tai150a problem, whereas minimal or even debatable superiority could be observed on the M-n151-k12 and CMT11 problems). As opposed to what was observed for TSP, diverse values of α and β were selected for different scenarios. What is more, ExpRnd did not perform that well; however, newly proposed ExpRndMax exhibited better performance. The Expected variant again showed unstable performance across the test instances; however, it gave the best results (on average) of all of the configurations for the CMT11 problem, for example. The Pairing variant was represented by two configurations (which emerged from the Pairing scenario, but also from the TwoDimAll scenario). These often gave similar results (except for the tai150c problem, where one of them could be considered to be the best and the other the worst of the tested configurations). Overall, the configurations that were found by *irace* and their qualitative results on the test instances were different from those that were obtained for TSP (see 4.2). Still, the Pairing variant looked the most promising here as well; this was also due to the fact that it was selected in the scenario with an unspecified interpretation type.

As for the execution time of the algorithm, the configuration from the Pairing scenario was only 10% slower than BasicSingle. This suggests that the overhead of the two-dimensional pheromone becomes lower when solving more-complicated problem types, as more computational resources are utilized for the logic of the problem itself (e.g., tracking capacity); therefore, the core metaheuristic algorithm took proportionally less time.

5. Conclusion

In this article, we have presented and evaluated a modification of the popular ant colony optimization algorithm – a two-dimensional pheromone structure. This was first proposed in [20], and our research represents a significant extension of this work – we have both introduced new variants

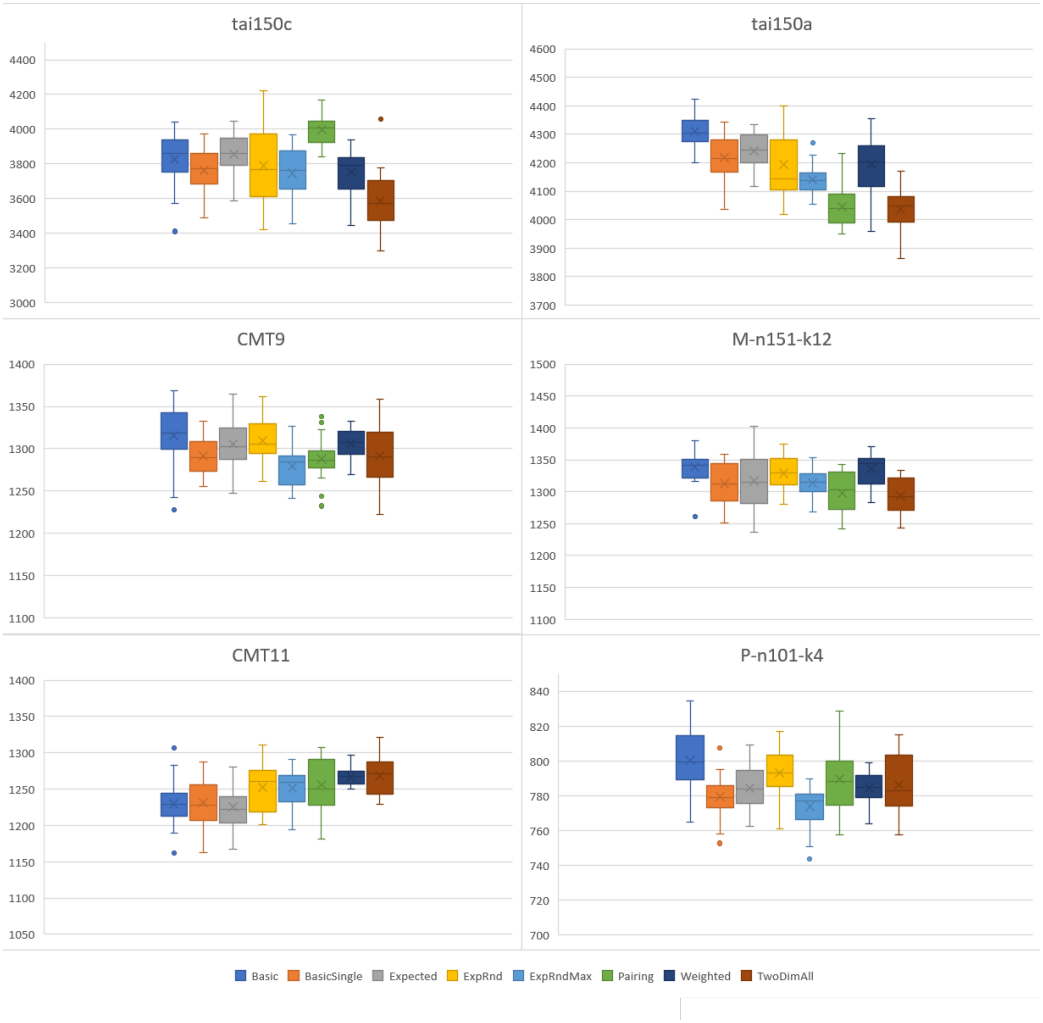

Figure 4: Results for *irace* VRP experiment

of the pheromone's behavior and broadened our conducted experiments in order to assess the idea.

The results of the experiments on multiple TSP and VRP problems that are presented in this article demonstrate that the ant colony optimization algorithm can benefit from the introduction of a more capacious model of the pheromone. In the selective experiments 4.1, a series of configurations with a two-dimensional pheromone overpowered those with a standard pheromone. Further experiments that were aimed at automatic configuration were performed with *irace*. Looking at the results, we can see multiple examples of superiority of the presented idea over the original ant colony optimization algorithm with a simple pheromone structure (even though it is difficult to point to a single best setup of the algorithm with a two-dimensional pheromone, which is in line with [21]). The most popular (i.e., the best-performing) variants of the ACO algorithm use just a single top solution in each iteration from which to learn by updating its pheromone. Having multiple values instead of a single one for each edge enables the algorithm to extract knowledge from more solutions and give better results as a result. As for the execution time of the algorithm, the selective experiments showed that the overhead of the two-dimensional pheromone can be mitigated by reducing the size of the colony while still achieving better results than the standard algorithm. At the same time, we observed that the top-performing configurations were only 10-15% slower than the standard version when using the same number of ants in the experiments with *irace*. The overhead is not discrediting (especially for those applications where minimizing the execution time is not crucial) and becomes lower for more-complicated problem types.

Having widely evaluated the idea of the two-dimensional pheromone on single-objective optimization problems, we will now aim at adjusting it to multi-objective optimization problems (this was actually our preliminary plan, with the idea of increasing the dimensionality of the pheromone in ACO). In such problems, there are normally a set of solutions (a so-called Pareto front) that can be treated as being the best instead of just a single one [22, 23, 24, 25]. Therefore, our intuition is that applying an extended model of the pheromone could turn out to be particularly effective in the optimization in multiple dimensions, since it can benefit from the ability to encode more information in a more complex pheromone structure.

Acknowledgements

This research was funded in whole or in part by National Science Centre, Poland, Grant no. 2021/41/N/ST6/01776 (GS). For the purpose of Open Access, the authors have applied a CC-BY public copyright license to any Author Accepted Manuscript (AAM) arising from this submission. The research presented in this paper has been also financially supported by: Polish Ministry of Science and Higher Education funds assigned to AGH University of Science and Technology (MKD) and ARTIQ project – NCN: DEC-2021/01/2/ST6/00004, NCBR: DWP/ARTIQ-I/426/2023 (AB).

References

[1] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: optimization by a colony of cooperating agents, IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 26 (1) (1996) 29–41.

[2] B. Bullnheimer, R. F. Hartl, C. Strauss, A new rank based version of the ant system. a computational study. (1997).

[3] T. Stützle, H. H. Hoos, Max–min ant system, Future generation computer systems 16 (8) (2000) 889–914.

[4] M. Dorigo, L. M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, IEEE Transactions on evolutionary computation 1 (1) (1997) 53–66.

[5] M. Starzec, G. Starzec, A. Byrski, W. Turek, Distributed ant colony optimization based on actor model, Parallel Computing 90 (2019) 102573.

[6] M. Starzec, G. Starzec, A. Byrski, W. Turek, K. Pietak, Desynchronization in distributed ant colony optimization in hpc environment, Future Generation Computer Systems 109 (2020) 125–133.

[7] C. García-Martínez, O. Cordón, F. Herrera, A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria tsp, European journal of operational research 180 (1) (2007) 116–148.

[8] D. Angus, C. Woodward, Multiple objective ant colony optimisation, Swarm intelligence 3 (2009) 69–85.

[9] J. G. Falcón-Cardona, G. Leguizamón, C. A. Coello Coello, M. G. Castillo Tapia, Multi-objective ant colony optimization: An updated review of approaches and applications, Advances in Machine Learning for Big Data Analysis (2022) 1–32.

[10] M. Dorigo, Optimization, learning and natural algorithms, Ph. D. Thesis, Politecnico di Milano (1992).

[11] M. Dorigo, G. Di Caro, Ant colony optimization: a new meta-heuristic, in: Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), Vol. 2, IEEE, 1999, pp. 1470–1477.

[12] N. Guo, B. Qian, J. Na, R. Hu, J.-L. Mao, A three-dimensional ant colony optimization algorithm for multi-compartment vehicle routing problem considering carbon emissions, Applied Soft Computing 127 (2022) 109326.

[13] M. López-Ibánez, J. Dubois-Lacoste, L. P. Cáceres, M. Birattari, T. Stützle, The irace package: Iterated racing for automatic algorithm configuration, Operations Research Perspectives 3 (2016) 43–58.

[14] K. van der Blom, S. Boonstra, H. Hofmeyer, T. Bäck, M. T. Emmerich, Configuring advanced evolutionary algorithms for multicriteria building spatial design optimisation, in: 2017 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2017, pp. 1803–1810.

[15] A. J. Nebro, M. López-Ibáñez, C. Barba-González, J. García-Nieto, Automatic configuration of nsga-ii with jmetal and irace, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, 2019, pp. 1374–1381.

[16] L. Pérez Cáceres, F. Pagnozzi, A. Franzin, T. Stützle, Automatic configuration of gcc using irace, in: Artificial Evolution: 13th International Conference, Évolution Artificielle, EA 2017, Paris, France, October 25–27, 2017, Revised Selected Papers 13, Springer, 2018, pp. 202–216.

[17] M. López-Ibáñez, T. Stützle, Automatic configuration of multi-objective aco algorithms, in: International Conference on Swarm Intelligence, Springer, 2010, pp. 95–106.

[18] M. Lopez-Ibanez, T. Stutzle, The automatic design of multiobjective ant colony optimization algorithms, IEEE Transactions on Evolutionary Computation 16 (6) (2012) 861–875.

[19] L. C. Bezerra, M. López-Ibánez, T. Stützle, Automatic generation of multi-objective aco algorithms for the bi-objective knapsack, in: International Conference on Swarm Intelligence, Springer, 2012, pp. 37–48.

[20] G. Starzec, M. Starzec, S. Bandyopadhyay, U. Maulik, L. Rutkowski, M. Kisiel-Dorohinicki, A. Byrski, Two-dimensional pheromone in ant colony optimization, in: International Conference on Computational Collective Intelligence, 2023, pp. 459–471.

[21] D. Wolpert, W. Macready, No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82. doi:10.1109/4235.585893.

[22] M. Fleischer, The measure of pareto optima applications to multi-objective metaheuristics, in: International conference on evolutionary multi-criterion optimization, Springer, 2003, pp. 519–533.

[23] I. Alaya, C. Solnon, K. Ghedira, Ant colony optimization for multi-objective optimization problems, in: 19th IEEE international conference on tools with artificial intelligence (ICTAI 2007), Vol. 1, IEEE, 2007, pp. 450–457.

[24] K. Deb, Multi-objective optimisation using evolutionary algorithms: an introduction, in: Multi-objective evolutionary optimisation for product design and manufacturing, Springer, 2011, pp. 3–34.

[25] T. M. Deist, M. Grewal, F. J. Dankers, T. Alderliesten, P. A. Bosman, Multi-objective learning to predict pareto fronts using hypervolume maximization, arXiv preprint arXiv:2102.04523 (2021).