Noname manuscript No.
(will be inserted by the editor)

PolyACO+: A Multi-Level Polygon-based Ant Colony Optimisation Classifier

Morten Goodwin · Torry Tufteland · Guro Ødesneltvedt · Anis Yazidi

the date of receipt and acceptance should be inserted later

Abstract Ant Colony Optimisation for classification has mostly been limited to rule based approaches where artificial ants walk on datasets in order to extract rules from the trends in the data, and hybrid approaches which attempt to boost the performance of existing classifiers through guided feature reductions or parameter optimisations. A recent notable example that is distinct from the mainstream approaches is PolyACO, which is a proof of concept polygon-based classifier that resorts to ant colony optimisation as a technique to create multi-edged polygons as class separators. Despite possessing some promise, PolyACO has some significant limitations, most notably, the fact of supporting classification of only two classes, including two features per class. This paper introduces PolyACO+, which is an extension of PolyACO in three significant ways: (1) PolyACO+ supports classifying multiple classes, (2) PolyACO+ supports polygons in multiple dimensions enabling classification with more than two features, and (3) PolyACO+ substantially reduces the training time compared to PolyACO by using the concept of multi-leveling. This paper empirically demonstrates that these updates improve the algorithm to such a degree that it becomes comparable to state-of-the-art techniques such as SVM, Neural Networks, and AntMiner+.

Keywords Ant Colony Optimisation · Classification · Polygon · Multi-leveling

A preliminary version of this paper (Goodwin and Yazidi, 2016) can be found in the Proceedings of the 10th International Conference on Swarm Intelligence, ANTS 2016. Part of this work has also been published as a Master's Thesis at University of Agder, Norway, Spring of 2016.

M. Goodwin · T. Tufteland · G. Ødesneltvedt
Department of Computer Science, University of Agder, Oslo, Norway

A. Yazidi
Department of Computer Science, Oslo and Akershus University College of Applied Sciences, Norway
E-mail: anis.yazidi@hioa.no

1 Introduction

Classification is the problem of predicting categories of unknown items on the basis of training data, and it is very common in machine learning with important application areas. Some well known examples include predicting sentiments of sentences, pinpointing objects in images, detecting patient deaths, and predicting the best move in Go. Hundreds of papers are published on the topic each year, which has resulted in a myriad of classification algorithms differing in principle, implementation, and performance. Classification becomes intrinsically challenging whenever the data to be classified is not easily separable in the feature space (Caruana et al, 2008; Madjarov et al, 2012).

Some of the best known classification techniques, such as Support Vector Machine (SVM) and perceptron-based classifiers, rely upon constructing mathematical functions having weights that efficiently separate two or more classes of data in the feature space. In two dimensional spaces, the separation boundary might be nonlinear and thus the decision boundaries might be complex. SVM deals with this situation by either projecting the data on a higher dimensional space or using a "kernel trick", which provides a separator not limited to a linear or polynomial function. The adoption of a kernel is equivalent to transposing the data to many dimensions, but the accuracy depends on the right choice of the kernel functions as well as on several other parameters. The latter choice is usually performed through manual trial and error.

The training process of a classifier can be considered as an optimisation task, and Ant Colony Optimisation (ACO) specifically has been proposed for training classifiers in three general areas. Firstly, ACO is commonly applied as a method to enhance state-of-the-art classifiers through parameter optimisations (Abadeh et al, 2008; Daly et al, 2009, 2011; De Campos et al, 2008; Jun-Zhong et al, 2009; Sharma et al, 2012). Secondly, some prominent studies, including AntMiner and AntMiner+ have used ACO as a rule based classifier (Martens et al, 2007). Thirdly, some recent advances have introduced PolyACO, a polygon based classification algorithm (Goodwin and Yazidi, 2016; Tufteland et al, 2016)[1]. Interestingly, PolyACO deals with the classification problem in a completely different manner from existing classifiers. Instead of relying upon mathematical functions, PolyACO surrounds the classes with polygons guided by artificial ants and ray casting.

This paper introduces PolyACO+, an extension of the existing PolyACO algorithm. PolyACO+ includes two main enhancements. Firstly, it is extended through its ability to classify multiple classes. PolyACO applies one polygon per problem, whereas PolyACO+ applies one polygon per class. This approach is similar to how an SVM deals with multiple classes. Secondly, PolyACO+ can handle datasets with more than two features. Since most datasets have multiple features, this enhancement makes PolyACO+ more suitable for real classification problems. This is achieved by taking majority votes on multiple two-feature projections, i.e., by invoking PolyACO+ on various 2-D spaces.

[1] Published by the authors of this paper.

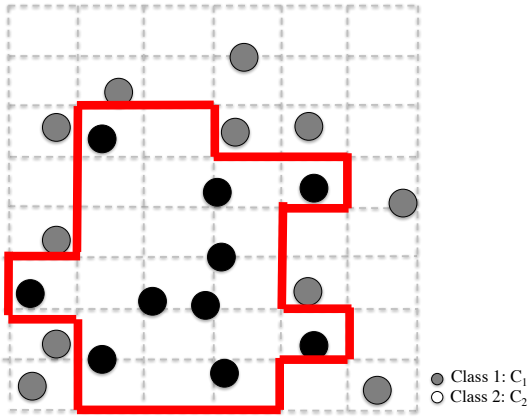


Fig. 1: Example of a simple two-class classification scenario with the classes Black (C_1) and Gray (C_2), each with two features.

The number of planes for n dimensions is calculated by counting the combinations of dimension pairs using the binomial coefficient: $\binom{n}{2}$. The ants walk along several two-dimensional planes, and in each plane the ants construct a polygon per class. Figure 1 shows a polygon constructed by PolyACO+ for the class C_1 in a two-dimensional plane. Furthermore, for performance reasons, the PolyACO+ reward function is designed to support a parallel architecture optimised to run on GPUs. These improvements make PolyACO+ dramatically faster and more accurate than its predecessor. Empirical results show that PolyACO+ performs similarly or better than other state-of-the-art classification algorithms in several classification tasks in terms of classification accuracy. Nevertheless, despite PolyACO+'s very good performance, its major hurdle is rather the long training time.

As for other classifiers, PolyACO+ has a training phase and a classification phase. The former's aim is to create polygons that encircle classes of items so that the polygons separate the training classes from each other. In this phase PolyACO+ finds a polygon s_j* per class C_j per pair of dimensions, consisting of vertices and edges. PolyACO+ maximises a function that measures how well the polygon s_j separates the items of class C_j from the others during the training phase. Thus, formally speaking, we aim to find an $s_j* \in \mathbf{S}$ so that $f(s_j*) \geq f(s_j)$ for each class C_j per pair of dimensions, where $\mathbf{S}$ consists of all possible polygons and the function $f(s_j)$ measures how well polygon s_j separates the data . The aim in the classification phase is to use the polygons as a basis to determine to which class a new unknown item to be classified belongs. The classification determines whether the item to be classified is within or outside of the polygon s, for each dimension. The overall classification result is a combination of classifications in all dimensions.

The paper is organised as follows. Section 2 presents the state-of-the-art in ACO based classification. Section 3 introduces PolyACO+. Section 4 presents

the results from applying PolyACO+ to classification problems and compares the results with state-of-the-art classifiers. Finally, Section 5 concludes and presents further work to be completed in this field.

2 State-of-the-art

In order to place the work into the correct context, this section presents different state-of-the-art algorithms for classification, placing an emphasis on classification using ACO.

2.1 Rule discovery classification

Rule discovery is a data mining task that generates a set of rules describing each class or category in a dataset. Based on labeled data, the algorithm defines a set of rules. The goal is to make predictions about unknown data using IF <conditions> THEN <class> rules, where <conditions> is constructed by terms in the form of (term1 AND term2 AND...).

From a historical perspective, the first application that used ACO for classification was AntMiner (Parpinelli et al, 2002). AntMiner is an algorithm that uses artificial ants for to discover classification rules. Several improvements to AntMiner have been suggested through the years (Liu et al, 2003), of which one very successful example is AntMiner+ (Martens et al, 2007).

AntMiner+ is an extension of AntMiner, which includes modifications such as a directed acyclic graph to create the environment on which the ants move (Martens et al, 2007, 2011). It also uses $\mathcal{MAX}$-$\mathcal{MIN}$ Ant System ($\mathcal{MM}$AS) (Stützle and Hoos, 2000) to manage ant behaviour and pheromones, and has an *early stopping* criterion (discussed further below).

AntMiner+ starts by creating a directed acyclic graph environment. An ant starts in the *Start* vertex and stops at the *Stop* vertex. The resulting path represents a potential rule. The paths that are walked by most ants, according to a predetermined threshold, are kept as classification rules. Similar to $\mathcal{MM}$AS, only the ants that achieve the globally highest score update the pheromones, and all values are adjusted to stay within the boundaries τ_{max} and τ_{min}. The algorithm converges when one path reaches τ_{max} and all others are equal to τ_{min}. Subsequently, the rule associated with the path containing τ_{max} is extracted along with the training data covered by it. Ants are continuously released until a stop condition is reached. This can either be an early stopping criterion or the fact that none of the ants is able to extract a rule that covers at least one training point. If the latter happens, no rule can be extracted as all the paths have zero quality. This is typically caused by noise in the remaining data, which indicates that further rule induction is useless.

In contrast, AntMiner+ resorts to early stopping to avoid over-fitting. An early stop happens when the error measure on the validation set (one set which is 1/3 of the training set) starts to increase. Training is then stopped,

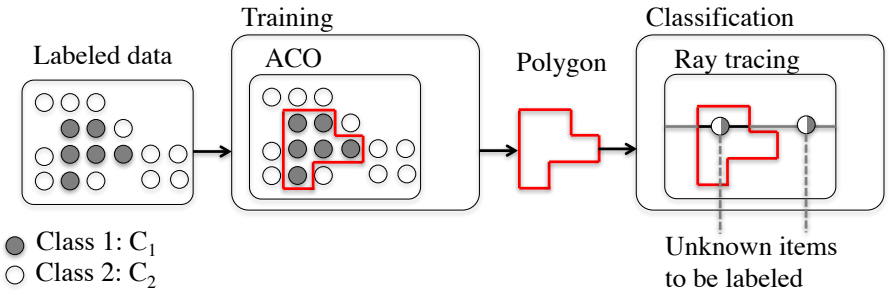


Fig. 2: Overview of training and classification in PolyACO. In the training phase, ACO is used to create a polygon that separates the classes, and items are classified based on whether they are inside or outside of the polygon. Classification happens through ray tracing where geometric rays are cast at the y-value of items to determine if the polygon surrounds the items.

thus effectively preventing the rule from fitting the training data noise. It should be noted that early stopping causes loss of data that cannot be used for construction rules, and is therefore better fitted for larger datasets.

In addition to the previously mentioned AntMiner series including its variations (Aribarg et al, 2012; Martens et al, 2007, 2011; Tripathy et al, 2013), the literature also includes other ACO rule-based classifiers. Perhaps the most notable example is Ant-labeler, a semi-supervised method for assigning labels to unlabeled data (Albinati et al, 2015). It uses ACO as a learning method and, during a self-training process, generates rule-based models. This results in a pheromone matrix from which classification rules are derived.

2.2 PolyACO

PolyACO is a grid-based polygon algorithm aimed at creating boundaries by surrounding and separating classes guided by ACO (Goodwin and Yazidi, 2016; Tufteland et al, 2016). Figure 2 presents an overview included here for explanatory purposes. The figure is an excerpt from the original PolyACO paper (Goodwin and Yazidi, 2016) and shows an example of how PolyACO is trained for the two classes C_1 and C_2 by surrounding items from only C_1 with a polygon. A similar approach has been proposed using learning automata in Goodwin et al (2016).

PolyACO is a rudimentary classification algorithm that only supports classification of two classes at once and only classes that have two features. The reason for this is that in PolyACO the ants explore solutions in a grid-like graph environment that is generated from the training data. Therefore, PolyACO is limited to solving classification problems in two dimensional spaces.

Ants are released sequentially with random initial positions. The ants explore and find paths in a similar manner to traditional ACO for path finding.

Instead of finding a path from a source to a goal, they end up at the same position from which they started. When ants return to their original position, their travelled path will have formed a polygon shape. To determine path quality PolyACO uses a combination of the polygon perimeter and a score of how well the training data is positioned relative to the polygon. This quality measurement is used as the reward function, and the objective of the algorithm is to maximize it.

After each ant walk, the pheromone trail is evaporated to avoid stagnation. Evaporation on an edge sets the edge to the minimum pheromone value, and is applied to each edge with the probability ρ called evaporation rate. For example with an evaporation rate of 0.01, a random sample of 1% of the edges will have their pheromone value reset to the minimum pheromone value after each ant walk.

Ants only deposit pheromones on the edges if their solution is better than the global best solution. Additionally, the global best solution is reinforced after each iteration to save it from gradually evaporating. This is in line with the approach used in $\mathcal{MM}$AS (Stützle and Hoos, 2000). Figure 3 shows an example of how the best known polygon evolves over multiple ant walks.

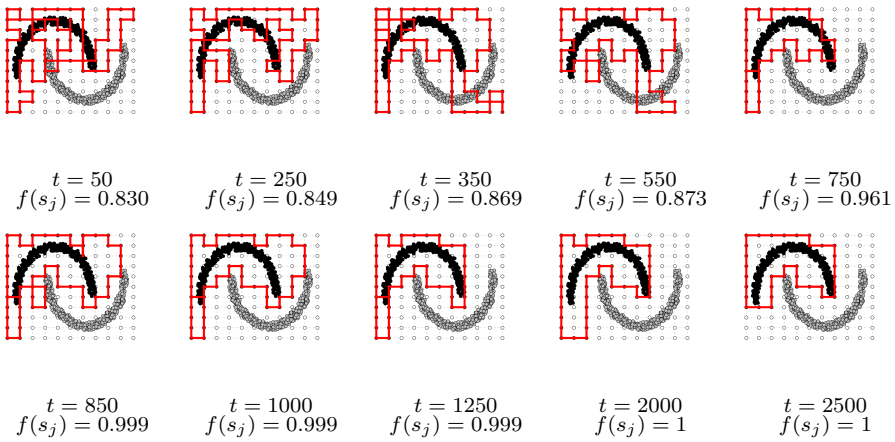
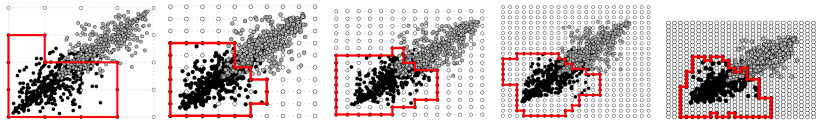


Fig. 3: Example of best known polygon s_j* over training periods.

The environment where the ants explore solutions is constructed based on the training data. It is a squared bi-directional weighted graph where all edges along a given axis are of equal length, thus forming a grid-like environment. The limits of the graph along axis k, G^k_{min} and G^k_{max}, are initialised with the maximum and minimum values of the data points along each axis. A minor value ϵ is added to the max value and subtracted from the min value in order to encapsulate all data points within the graph. Formally:

$$G^k_{min} = min(T^k_{ij}) - \epsilon \tag{1}$$



μ=5 μ=10 μ=15 μ=20 μ=25

$f(s_j) = 0.826$ $f(s_j) = 0.831$ $f(s_J) = 0.828$ $f(s_j) = 0.840$ $f(s_j) = 0.832$

Fig. 4: Example of best known polygon s_j* for varying granularity factor μ. The parameter μ can be adjusted by the user.

$$G^k_{max} = max(T^k_{ij}) + \epsilon \tag{2}$$

where T^k_{ij} represents an edge from node i to node j at axis k.

The resolution of the graph environment can be manually adjusted through a granularity factor μ provided to the grid at initialisation. $\mu > 2$ is an integer that gives the granularity of the grid along both axes. For example, a μ value of 5 would result in a 5x5 grid, a value of 10 would result in a 10x10 grid, etc. Since the graph is a square the resolution is the same for all axes. In other words μ determines the granularity of the environment.

While a high granularity gives the ants more paths to choose from and the possibility to create more accurate solutions, it also increases the size of the search space and the time it takes for a single ant to complete its path. Figure 4 shows an example demonstrating that more fine-grained graphs can construct better classifiers than more coarse-grained graphs when given identical datasets.

After every completed ant walk, the pheromones in the graph environment are updated. The amount of pheromone laid in an area of the graph depends on the quality of the ant solution. The aim is to create a polygon s_j that surrounds all items of class C_j correctly and does not surround any item of the opposite classes. The quality of a solution s_j is measured by a reward function $f(s_j)$ which is a function of the perimeter of the polygon and the number of elements that are correctly placed within it:

$$f(s_j) = \frac{\sum_{t_i \in C} h(t_i, s_j)}{|C|}, \tag{3}$$

where $h(t_i, s_j)$ is a function that determines whether an element t_i is on the inside of the solution s and C contains all items to be classified. PolyACO uses ray casting to determine if an element is on the inside or outside of a solution. $h(t_i, s_j)$ is defined as follows:

$$h(t_i, s_j) = \begin{cases} 1 & \text{if } t_i \in C_j \text{ and is inside of } s_j \\ 1 & \text{if } t_i \notin C_j \text{ and is outside of } s_j \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where C_j is the class represented by polygon (solution) s_j so that a perfect solution surrounds all items of the class C_j and no items of opposite classes.

In other terms, for an item that truly belongs to class C_j according to its label and which is inside the polygon s_j, it will get $h(t_i, s_j) = 1$. For an item that does not belong to class C_j according to its label and which falls outside the polygon s_j, it will get $h(t_j, s_j) = 0$. The function $f(s_j)$ includes $h(t_i, s_j)$ for all items and thus reflects the homogenity of items inside polygon s_j.

To avoid overly complex polygons, we should favor polygons with smaller perimeters. This is achieved by modifying the reward function to be proportional to the inverse of the perimeter. Therefore, the smaller the perimeter, the higher the reward. The pheromone update can then be summarised utilising the following equations:

$$\tau_{ij} \leftarrow \left[\tau_{ij} + \Delta\tau_{ij}^{best}\right]_{\tau_{min}}^{\tau_{max}} \tag{5}$$

$$\Delta\tau_{i,j}^{best} = \frac{f(s_j)}{|s_j|} \tag{6}$$

While this update form is similar to how pheromones are updated in $\mathcal{MMAS}$, it includes factoring in the new reward function.

The current version of PolyACO has some weaknesses compared to other state-of-the-art classifiers. First, it is unable to classify data in more than two dimensions. Secondly, it does not handle classification problems comprising more than two classes. Thirdly, it is very slow and takes a long time to train.

2.3 Multi-leveling

Multi-leveling is a technique commonly applied to combinatorial optimisation problems as a way to efficiently search for solutions in a complex space. It involves recursive coarsening of complex problems to obtain a hierarchy of approximations to the original problem. An initial solution is found and then refined at each level as the problem space is coarsened (Lian et al, 2015; Walshaw, 2004).

Multigrid methods are a class of multi-level algorithms originally developed to solve boundary-value partial differential equations in geometric domains (Brandt, 1977). Subsequently, they have proven useful to solving several kinds of geometrically based optimisation problems. The idea is to construct a sequence of Cartesian grids, where each grid is typically twice as coarse as the former (Brandt, 1988). An initial solution is first obtained at the coarsest level, then refined by moving up and down the hierarchy of grids according to some heuristic.

Adaptive Mesh Refinement (AMR) is a multigrid technique used to dynamically modify the grid during computation by increasing the resolution of the grid in areas of interest (Berger and Colella, 1989). The strategy for selecting the grid areas in which to increase the resolution depends on the problem.

AMR makes it possible to solve certain problems with a much higher precision level than traditional multigrid methods as it requires less computing power than a uniform high-resolution grid. AMR has, for instance, been used to model the collapse and fragmentation of molecular clouds with an unprecedented accuracy (Klein, 1999).

2.4 Other classification schemes using Ant Colony Optimisation

Several other classification algorithms using ACO and other metaheuristic algorithms are available in the literature.

Salama and Abdelbar (2016) use a cluster based classification approach with ACO. They introduce a two-step approach. First, they assign a class to a cluster using ACO. Subsequently, they continue using a local classifier which is independent of ACO. The approach introduces instance- and medoid-based ACO clustering and is basically an optimiser for existing classifiers.

The same authors introduce an approach for learning neural network structures using ACO (Salama and Abdelbar, 2015). Accordingly, they propose ANN-Miner to learn the structure of a feed-forward network, which in turn can be used to predict unknown classes of new patterns.

Varma et. al (2015) introduce NRSACO as a method for setting attribute reduction as an extension of rough sets theory. They claim that in contrast to standard rough sets, NRSACO is able to avoid being stuck in local minima. This is similar to feature selection mechanisms using particle swarm optimisation, which aim at reducing the number of features for classification so that the classification becomes faster and easier (Xue et al, 2014); and to how ACO is used for data reduction (Salama and Abdelbar, 2016).

An approach using ACO to optimise decision boundaries in decision trees is introduced in (Sapin et al, 2015). Specifically, ACO is used to find nucleotide polymorphisms, which are then combined into a decision tree.

3 PolyACO+

PolyACO+ is an improvement of PolyACO which can handle an arbitrary number of dimensions and classes. It also drastically reduces the training time by computing the reward function in parallel on GPUs and by employing a dynamic multi-level scheme. This section describes the working of PolyACO+.[2]

3.1 Multiple-class classification

Figure 5 presents an overview of how PolyACO+ handles multiple classes. For example, when the number of classes is two, one polygon is sufficient to create

[2] The full source code of PolyACO+ may be found at `https://github.com/UIA-CAIR/PolyACOPlus`

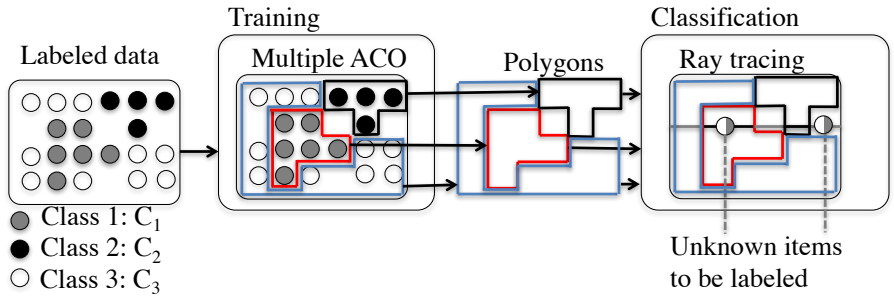


Fig. 5: Overview of training and classification for PolyACO+ with multiple classes.

a decision boundary. However, when the number of classes is larger than two, PolyACO+ creates one polygon per class. Since there are multiple polygons, an item to be classified belongs to three possible cases:

1. **The item is located inside one polygon**; in this case, the item is simply classified as the polygon to which it belongs.
2. **The item is not located within any polygons**; in this case, it is not possible to determine to which class it should belongs. This problem is solved by simply accepting "no class" as a valid output from the classifier.
3. **The item is encapsulated by several polygons**; PolyACO+ handles this situation by randomly selecting to which class the item should belong. However, one consequence is that PolyACO+ classification becomes stochastic. An alternate option could therefore be to handle class conflicts by order of precedence. For example, given that the polygons from class C_1 and C_2 both surround a sample, then the sample is to be classified as C_1 since C_1 has the lower index value. This action would produce an unjustified bias towards polygons with low index values and might yield unexpected results.

3.2 Parallelisation with GPU

Based on recent advances from Tufteland et. al (2016), we now introduce GPU in the training phase. This is done by parallelising the most costly part of the algorithm, the reward function, to the graphics processing unit (GPU) using CUDA (Ryoo et al, 2008).

The $h(t_i, s)$ function combines all data points and edges in a training set. This set can be parallelised, which makes it well suited for a GPU kernel function. In practice, the GPU is invoked once per ant, and returns a two-dimensional array of points in the training set and the number of edges produced by the ant. For more details, we refer the reader to (Tufteland et al, 2016).

The two elements required for classification are the trained model and an implementation of the classification phase. The trained model in PolyACO+ is simply a description of the polygons constructed in the training phase. Although constructing these polygons is very computationally expensive, once they have been constructed they can be reused in the classification phase. This phase is much less computationally expensive than the training phase, and can therefore be implemented on other devices relatively easily.

3.3 Multiple features

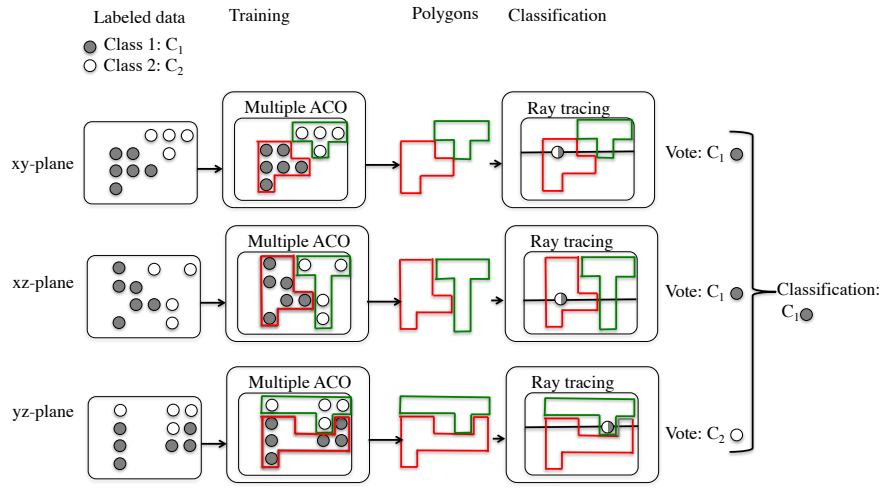


Fig. 6: Overview of training and classification for PolyACO+ with several features.

PolyACO+ supports multiple features by splitting a multi-feature classification problem into several two-dimensional sub-problems which are trained independently. The overall classification is a combination of the results from all sub-problems through a majority voting scheme. More precisely, the overall class prediction is derived by taking the most common class prediction from all the sub-problems, as illustrated in Figure 6.

3.3.1 Training

Instead of constructing solutions for only one plane as in the case of PolyACO, PolyACO+ constructs solutions in all the planes that the dataset consists of, and handles each plane individually. The number of possible planes depends on the number of features in the dataset. For example, a three-dimensional

feature space with axes x, y and z has three planes xy, xz and yz (See Figure 6). More generally, the number of planes for an n dimensional feature space is simply equal to the number of dimension pairs and is given by: $\binom{n}{2}$[3].

Further, the training phase constructs one polygon for each class in the dataset per plane. Thus, for v classes and n dimensions, the total number of polygons created by PolyACO+ is

$$\binom{n}{2} \times v \tag{7}$$

For example, for a dataset with 3 classes and 4 dimensions, the number of polygons in the training model is $\binom{4}{2} \times 3 = 18$.

3.3.2 Classification

PolyACO is a two-class classifier. PolyACO+ uses the concept of majority voting to extend PolyACO for handling multi-class problems. In order to classify a sample, each class is awarded a vote if it surrounds a sample in a given plane. For example, a class is awarded 2 votes if the polygons surround a sample in 2 different planes. The sample is classified as the class with the most votes when all votes are counted.

More formally, let t_i be an unlabelled item to be classified, $h(t_i, s)$ is computed according to Equation 4:

$$h(t_i, s) = \begin{cases} 1 & \text{if } t_i \text{ is inside of } s \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

The function $v(t_i, k)$ counts the votes for a given sample t_i relative to the class k

$$v(t_i, k) = \sum_{i=0}^{p} h(t_i, s_{k,j}) \tag{9}$$

where $s_{k,j}$ is the polygon solution belonging to class k and plane j, and p is the total number of planes (see Equation 7). The votes for all classes k are gathered in a vector $\mathbf{v_{ti}} = (v(t_i, 0), v(t_i, 1), \ldots v(t_i, k))^T$. Finally, the predicted class for sample t_i is defined as

$$p(t_i) = argmax(\mathbf{v_{ti}}) \tag{10}$$

where $argmax(\mathbf{v_{ti}})$ is a function that returns the index of the largest element from vector $\mathbf{v_{ti}}$.

In layman's terms, PolyACO+ decides which class the item should belong by counting the number of polygons from each class surrounding the item.

[3] Inevitably, the number of planes grows exponentially with the number of features. However, feature selection and reduction methods could be used to deal with this problem. See section 5.

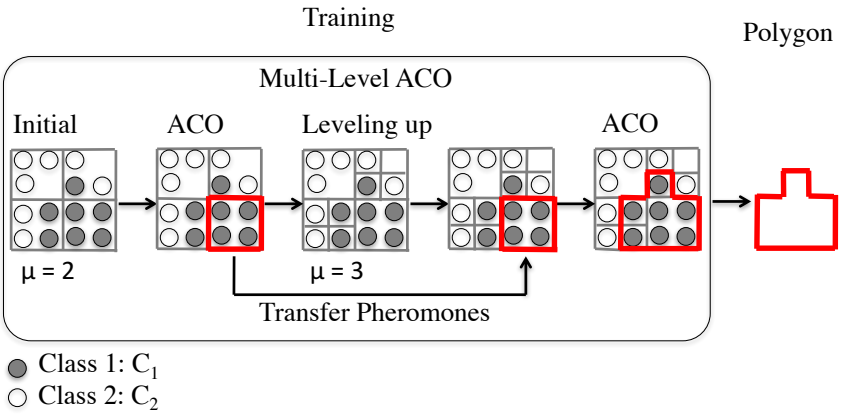


Fig. 7: Overview of multi-leveling in PolyACO+ with dynamic AMR.

3.4 Multi-leveling

PolyACO+ is enhanced with multi-leveling capability and mesh refinement, which removes the need for manually tuning the granularity values of the grid. Following the principles of mesh refinement, the graph starts with a low granularity, for example $\mu = 3$, and adaptively increases the granularity. The rise in leveling happens after the ants have converged onto a path according to the stop criterion, i.e., when no new solution is found for a fixed number of iterations η. The pheromone trail is transferred over to the new graph, giving the ants an indication as to where the good paths are. The granularity is increased until a given maximum level M is reached. At this point, when the ants converge, a solution is found. Figure 7 shows a concrete example of multi-level PolyACO+ from $\mu = 2$ up to $\mu = 3$.

We present two alternative approaches to multi-leveling in PolyACO+: a naïve multi-level approach based on traditional multigrid techniques and a more sophisticated approach with adaptive mesh refinement (AMR) that intelligently selects in which part of the graph the granularity should increase.

3.4.1 Naïve multi-level

In the naïve multi-level approach, the grid is constructed initially with a low granularity and is "leveled up" by increasing the granularity on the entire grid when the ants converge. The convergence rate for a level is defined when η ants have not found an improved solution. After η ants have walked without improvement, PolyACO+ levels up by splitting each edge into two new edges, and new edges and vertices are created in order to connect all vertices. During the experiments, the convergence rate η is set to 800.

Figure 8 shows the results of the experiment where we compare the results from a naïve multi-level run and a set of fixed granularity values. PolyACO+ with a μ value (grid size) of 5 converges rapidly to one solution, the found solution has the lowest score compared to the rest of the solutions. When we set $\mu = 10$, we see that the convergence speed decreases, but the quality of the solution increases. The case with $\mu = 15$ yields an even slower convergence speed, but scores the highest of the three. Nonetheless, all solutions obtained by the fixed granularity approach are surpassed by the naïve multi-level approach in both score and convergence time. The figure also shows that the multi-leveling PolyACO+ produces better solutions after 300 seconds, long after the other approaches have converged. The full test results are presented in Table 1, where we have also listed results obtained with μ values of 30 and 60.

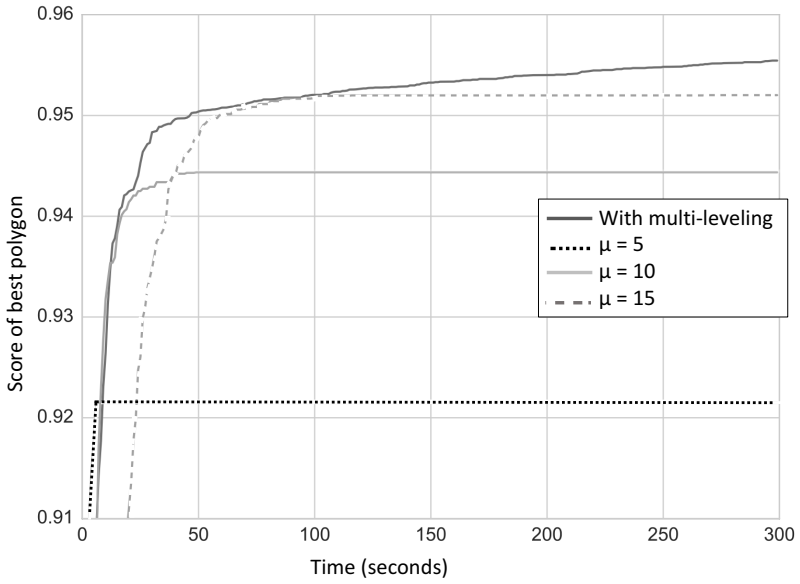


Fig. 8: Naïve multi-level (average of 20 runs).

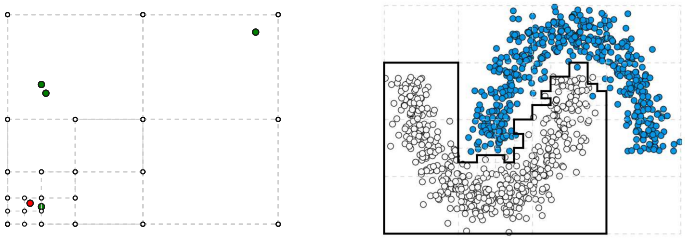
3.4.2 Multi-level with adaptive mesh refinement (AMR)

The naïve multi-level approach yields both increased accuracy and faster convergence speed. However, this comes at the cost of creating unnecessarily many edges and vertices in areas where there is no training data. Those edges and vertices do not capture any points, and therefore will only increase the search

Granularity, μ	Score, 15 seconds	Score, 300 seconds
3 with multi-level	**94.25%**	**95.54%**
3	73.57%	73.57%
5	92.15%	92.15%
10	**94.14%**	94.44%
15	91.09%	**95.20%**
30	67.39%	93.00%
60	54.93%	69.07%

Table 1: Quality (Eq. 3) of naïve multi-level after 15 and 300 seconds for different granularity levels.

space and reduce the convergence speed. Instead of having a high granularity overall, a more desirable approach would be to have a higher level of granularity only in the areas that are covered by data points. This is achieved with multi-level AMR.



(a) (b)

Fig. 9: AMR multi-level PolyACO+ applied to: (a) a simple example and also to: (b) the semi circular dataset.

The local improvement works by recursively walking through the graph section by section. A section is defined as any four edges that forms a square. Since a section is also a closed polygon, ray casting is applied to determine if a point is located inside or outside the section. If a section has data points of both the target class and of any other class, the section is divided into four new subsections. Figures 9a and 9b show a multigrid with AMR on a simple constructed scenario, and a more complex semi-circular dataset. The figures show how the grid is fine-grained in areas with data from both classes, and coarsely grained in the remaining areas.

AMR can be applied to PolyACO+ in two ways. The first approach is a static AMR where it is applied only once on training data and the outputted grid remains unchanged during the entire training phase. The second is is a dynamic AMR where it is applied several times during training as illustrated in Figure 7. The dynamic approach starts with a coarser grid and then gradually increases the granularity over time. In this way, it can initially find good solutions very quickly, and then refine these solutions in the next levels. Both

the static and dynamic AMR approaches increase the granularity only in the most relevant parts of the grid. AMR approaches differ from the simpler naïve multi-level ones since the latter increase the granularity in the entire graph.

The dynamic AMR approach is used throughout this paper; all approaches are compared and discussed in Section 4.6.

Convergence The stopping criterion in PolyACO is specified by the number of ants to run before stopping. The obtained solution is given by the ant path that has achieved the best score when the stopping criterion is reached. In PolyACO+ convergence is defined as when no new best solution is found for a given number of ants η. The intuition behind this stopping criterion is that the algorithm will not terminate as long as it is still finding new and better solutions. The convergence is per level, and when the convergence rate is reached at the final level M, the algorithm terminates. Figure 10 illustrates the positive effects of the dynamic convergence in the multi-level approach. After each level up (the crosses), the rate at which the algorithm finds new better solutions increases immediately. More productive levels, i.e., levels where better solutions are found frequently, are assigned more ants than levels where no better solutions are found. Regarding the example in the figure, we can observe that level 3 and 4 are assigned more ants than level 5 and 6, because they are more productive.

3.5 Early stopping

In order to improve efficiency, an optional early stopping criterion is applied by terminating earlier if the best polygon score does not improve for a given number of levels. For example, if the max level M is set to 5 and early stopping convergence is set to 2 and the algorithm finds the best possible solution already at level 1, the algorithm will terminate at level 3 instead of level 5 for that particular polygon.

3.6 Multiple data points with shared coordinates

It is often the case that multiple data points share coordinates in a plane. For example, a two by two grid has only four possible coordinates. If the training set is larger than the number of possible coordinates, certain points must share coordinates. A problem occurs when the algorithm levels up: the granularity of a given section is increased if there are points of different classes within that section. In this manner, the grid is able to separate points of different classes without increasing the granularity on the entire grid. However, if points of different classes share coordinates, the multigrid is never able to separate these points, even with an infinitely high granularity. In order to handle this special case, PolyACO+ does not increase the granularity of a section if all points in the section share the same coordinate.

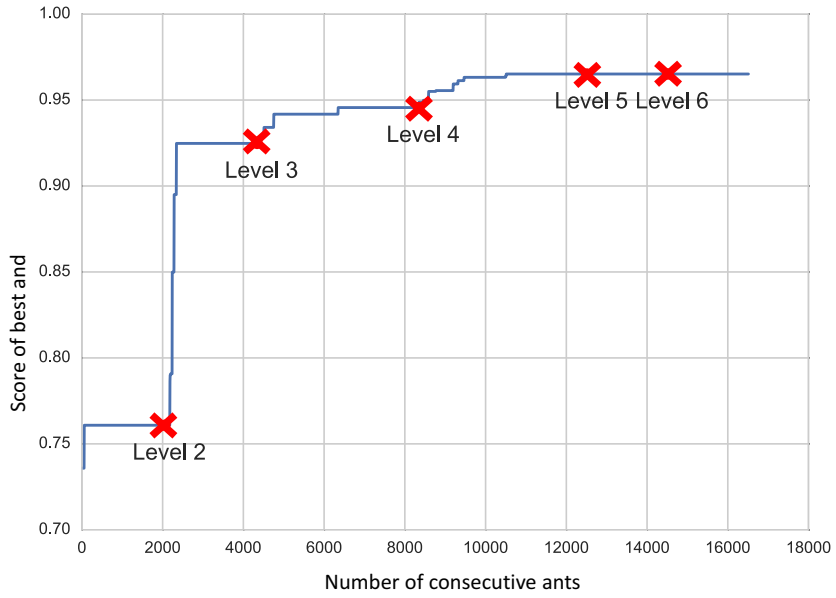


Fig. 10: Multi-level AMS convergence in construction of a single polygon ($\eta = 2000$, $M = 6$, average of 20 runs).

3.7 Additional enhancements

This section describes some additional enhancements to PolyACO+ compared to PolyACO. Equation 6 shows how pheromones are updated in PolyACO. This equation tends to bias the perimeter of the polygon compared to encapsulation of data points. This paper proposes an improved pheromone update equation using weights on the perimeter factor and data encapsulation respectively:

$$\Delta\tau_{i,j}^{best} = f(s_j)^{\alpha} \cdot \left(\frac{1}{|s_j|}\right)^{\beta} \tag{11}$$

For example, the significance of the length factor β can be decreased by setting it to a low value.

All ants are initialised with a random position in PolyACO. Consequently, this might cause many ants to start in a position that is far away from any data point in the target class, making it hard to find good polygons around the target clusters. Therefore, two new methods for selecting the start position, *Weighted* and *On_Global_Best*, are proposed. The *Weighted* method selects a start position based on the amount of placed pheromones. The *On_Global_Best* method selects the current global best solution. Both approaches produced su-

perior results compared to the random initialisation strategy. However, further investigation is needed in order to identify the best of the two approaches.

3.8 PolyACO+ parameters

Table 2 contains an overview of all the parameters of PolyACO+. The parameters ρ, τ_{min} and τ_{max} come from $\mathcal{MM}$AS, and α, β and M are introduced in PolyACO+.

Name	**Description**	**Default value**
τ_{min}	Minimum pheromone value	0.001
τ_{max}	Maximum pheromone value	1.0
ρ	Pheromone evaporation rate	0.02
η	Convergence rate	1200
M	Max granularity steps	6
α	Weight for the reward function	1.0
β	Weight for the polygon perimeters	0.01

Table 2: Overview of all algorithm parameters in PolyACO+

Unless stated otherwise, the default parameter values from Table 2 are used in all experiments throughout this paper. These were obtained by testing various parameter values over a given number of ants on the generated semi-circular dataset (see section 4.2.2) and then the best performing values were selected. This was done over several rounds for each parameter. For example, for the pheromone evaporation rate ρ we first tested the values 0.001, 0.01, 0.1 and 1.0 and the results have been calculated with an average over 20 runs per parameter. $\rho = 0.01$ performed the best. Therefore, we further tested the values 0.005, 0.02, 0.035 and 0.05. This time, $\rho = 0.02$ performed the best and was therefore selected as the default value for ρ.

4 Results

Experimental results are given in this section to demonstrate the efficiency of PolyACO+. We first start by presenting the synthetic and real data used in these experiments 4.1. We continue in section 4.2 by providing results from two-featured datasets (including challenging problems such as circular datasets), while section 4.3 is concerned with multiple-class classification with multiple attributes. Section 4.4 deals with fastening PolyACO+ using GPU and section 4.5 compares the results with other algorithms.

	sim	s-circ	over	circ	circ+ε
Instances	1000	1000	1000	1000	1000
Attributes	2	2	2	2	2
Polygons	1	1	1	1	1
Technique					
PolyACO+	100.0	100.0	85.20	100.0	94.80
PolyACO	100.0	100.0	85.20	100.0	94.80
Linear SVM	100.0	91.20	83.70	53.80	53.80
Polynomial SVM	100.0	99.70	82.20	89.20	77.80
Gaussian SVM	100.0	100.0	84.00	100.0	95.90

Table 3: Classification accuracy of PolyACO+ compared to state-of-the-art classification algorithms for the datasets simple Environment (sim), overlapping data (over), circular (circ), circular with noise (circ+ε), and semi-circular (s-circ).

4.1 Data

This section presents results from various scenarios ranging from simple classification problems using easily separable data to more complex settings involving both real-life and synthetic noisy data. For each generated scenario, 1000 data points per class are generated. For the real scenarios, the whole corresponding dataset is used. The real datasets used in the experiments are Iris, Breast Cancer Wisconsin (bcw) and Digits[4] from the UCI dataset repository (Lichman, 2013). In all cases, half of the data is used for training and the other half for classification. All scenarios are run with 10,000 ants unless otherwise explicitly specified.

4.2 Two dimensions and two classes

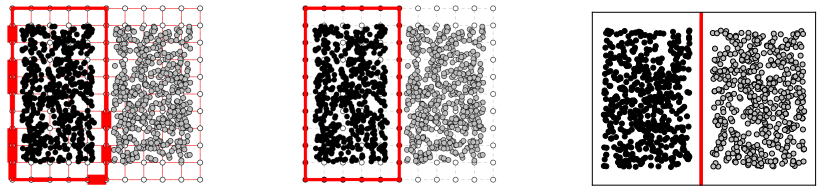
4.2.1 Simple environment dataset

This section presents a simple experimental setting as a proof of concept of PolyACO+ in two dimensions. The aim of the experience is to empirically demonstrate that the approach works in a simple environment containing two easily separable sets of data. The data is composed of two sets of points: C_1 and C_2. Figure 11a illustrates the pheromone trails at the end of the training phase. The thicker the line, the more pheromones are deposited there, which means that the algorithm is more certain that the edge is part of the best solution. From the figure, we observe that the pheromones have built a rectangular polygon encircling all items in C_1 without including any of the items in C_2. Figure 11b presents the best found polygon s based on the pheromone trail. Since this is a polygon that perfectly separates the classes, it yields $f(s_j) = 1$.

[4] Subset of "Pen-Based Recognition of Handwritten Digits Dataset (Lichman, 2013)" retrieved from scikit-learn (`http://scikit-learn.org/0.17/auto_examples/datasets/plot_digits_last_image.html`)

The mapping from pheromones to polygon in this example is quite straightforward. Lastly, for comparison purposes, Figure 11c depicts the corresponding linear SVM. It is interesting to observe that PolyACO+ and SVM are able to find the same boundaries.

This simplistic example indicates that when it comes to easily separable data, the result of the PolyACO+ is similar to that of a linear SVM. Table 3 shows an overview of the classification results. Both PolyACO+ and SVM reach an accuracy of 100.0 — which is not surprising given the simplicity of the classification task.



(a) PolyACO+ Pheromones (b) PolyACO+ Polygon (c) Linear SVM

Fig. 11: Example of classification of the simple environment dataset.

4.2.2 Semi-circular dataset

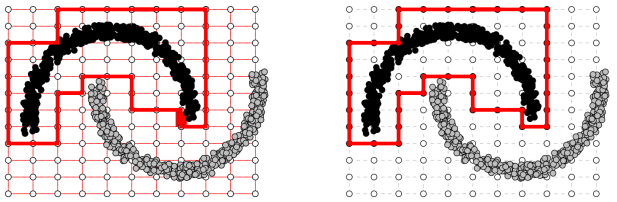
Figure 12 illustrates the behavior of the scheme in a more complex scenario involving semi-circles (or half moons) where there are no clear cut boundaries.

Despite the added complexity, the PolyACO+ approach works almost identically to the simple scenario in Figure 11; Figure 12a and 12b show the pheromone trail and polygon s respectively in the training data. We observe that there is an easy mapping from pheromones to polygon. Figure 12c shows the boundary found by a linear SVM, which is not perfect simply because the classification problem cannot be solved by a linear separator. Lastly, the polynomial SVM in Figure 12d produces better, but not perfect boundaries.

Table 3 shows that PolyACO+ achieves a classification accuracy of 100.0, while linear, polynomial and Gaussian SVM yield an accuracy of 91.2, 99.7 and 100.0 respectively.

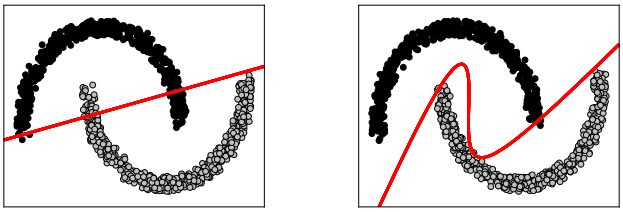
4.2.3 Overlapping dataset

In the above scenarios, the data is perfectly separable. In the current scenario, the data in Figure 13 is more challenging because it is overlapping and therefore no line or polygon can perfectly separate the datasets.



(a) PolyACO+ Pheromones

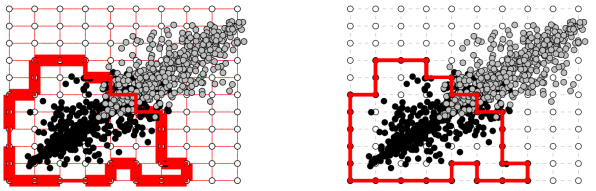
(b) PolyACO+ Polygon



(c) Linear SVM

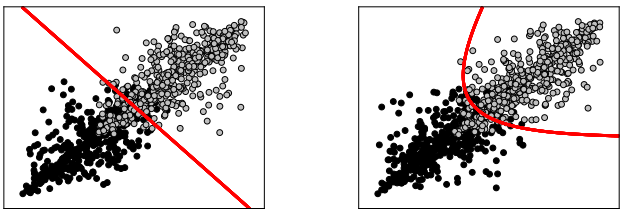
(d) Polynomial SVM

Fig. 12: Example of classification of the semi-circular dataset.



(a) PolyACO+ Pheromones

(b) PolyACO+ Polygon



(c) Linear SVM

(d) Polynomial SVM

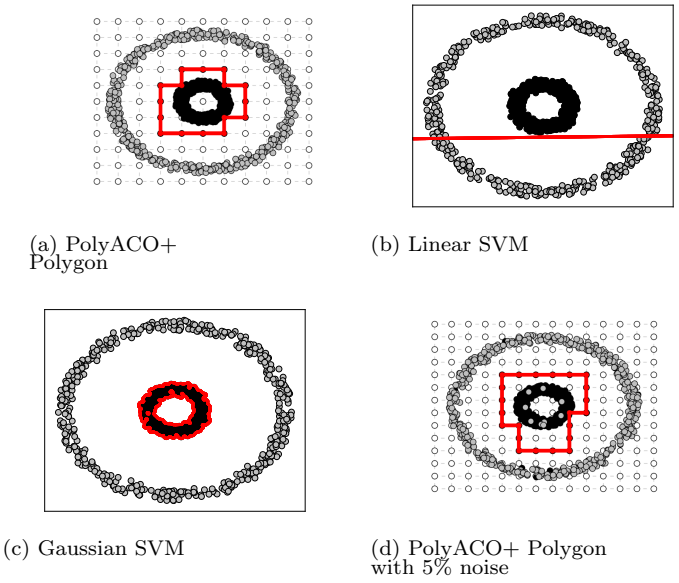
Fig. 13: Example of classification of the overlapping dataset.

Figure 13a shows the pheromones after the training phase. Concerning the left- and lower part of the polygon, the pheromone trail is strong and thus the lines are thick. In contrast, whenever the data overlaps, the scheme is less confident and the pheromone trail is weaker. This indicates that when the confidence of the classifier is strong, PolyACO+ provides strong pheromone trails. Figure 13b shows the corresponding polygon, and Figure 13c and 13d show corresponding boundaries of linear and polynomial SVM.

In Table 3, we observe that PolyACO+ reaches an accuracy of 85.2, while linear SVM reaches 83.7, and polynomial SVM reaches 82.2. One conclusion to be drawn from this example is that PolyACO+ finds a slightly better boundary than SVM, presumably because the rigged lines better fit the data than the straight and polynomial lines.

4.2.4 Circular dataset

The classification tasks when the data points of a class form a circular shape is particularly difficult without mapping it to multiple dimensions. The data is generated from a Gaussian distribution from two circles having the same center but with two different radius.



(a) PolyACO+ Polygon

(b) Linear SVM

(c) Gaussian SVM

(d) PolyACO+ Polygon with 5% noise

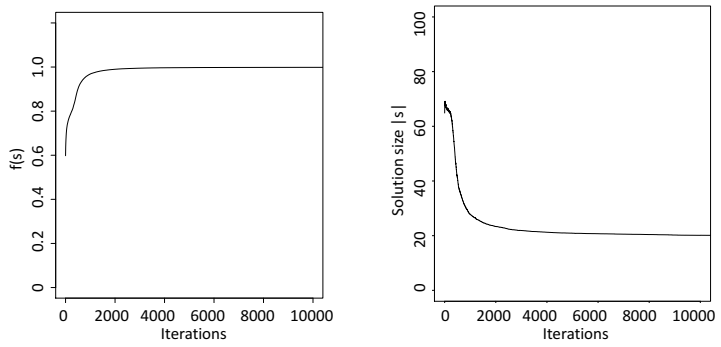
Fig. 14: Example of of classification of circles.

Figure 14a shows that the polygon is able to perfectly encircle class C_1, which is only matched by the SVM containing a Gaussian kernel in Figure

14c. The linear SVM in Figure 14b and the polynomial SVM (not presented as a figure) do not find any viable solution.

By adding 5% noise to the data, meaning that 5% of the data is intentionally wrongly labelled, Figure 14d shows that PolyACO+ is still able to obtain a nearly perfect solution.

Table 3 shows that PolyACO+ gets an accuracy of 1 compared to 53.8 for linear SVM and 89.2 for polynomial SVM. The PolyACO+ accuracy is only matched by Gaussian SVM. [5] In a noisy environment, the PolyACO+ algorithm has only marginally reduced level of accuracy, namely, 94.8. Correspondingly, the polynomial SVM accuracy dropped from 89.2 to 77.8 while the accuracy of Gaussian SVM dropped to 95.9. Figure 15 shows the evolution of the score for the best polygon s ($f(s_j)$) and of the size of the polygon ($|s|$). Pheromones are represented by the edges' width in the graph.



$f(s_j)$ over time Size of the solution over time — $|s_j|$

Fig. 15: Evolution of polygon over time. Average of 1000 runs.

4.3 Multiple classes and features

This section presents classification results with PolyACO+ with multiple classes and features.

4.3.1 Number of polygons with multiple classes

We have carried out many experiments on multiple class problems. We present results for two real datasets, Iris and bcw (discussed further in Section 4.5.1). Concerning this particular experiment, each dataset is run using three different

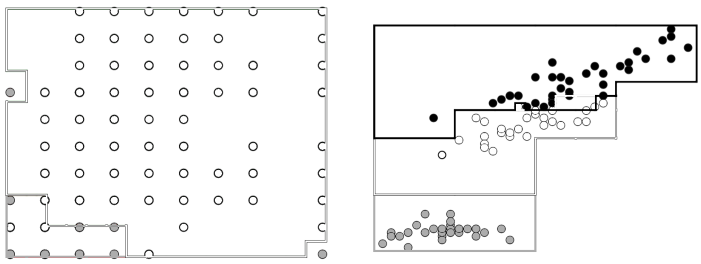
[5] Note that the choice of kernel, including the Gaussian SVM kernel, is not trivial and typically relies upon trial and error or expert knowledge of the field (Smola and Schölkopf, 2004). PolyACO+ has no such parameter to be tuned.

values for the convergence rate and 50-fold cross-validation. A summary of results with many other datasets is available in Section 4.5.2.

	Iris			**bcw**		
Convergence rate η	100	200	400	100	200	400
All-minus-one classes	90.40	93.20	93.84	97.28	97.03	96.79
All classes	92.44	94.12	95.28	96.92	96.92	97.12

Table 4: Classification accuracy levels using polygons for all and all-minus-one classes for different values of the stopping parameter η (100, 200 and 400). The stopping criterion takes place whenever a number of η ants have walked without any improvement.

The results in Table 4 demonstrate that while using all classes improves the classification accuracy levels on the Iris dataset, it does not produce the same effect on the bcw dataset. One possible cause for this discrepancy is that the data on the Iris dataset is continuous, while the bcw data is discrete. Figure 16 shows an example plane from the bcw dataset compared with an example plane from the Iris dataset. The data in the bcw dataset is more uniformly distributed over the entire plane than in Iris. Therefore, in this manner, the classifier constructs polygons that cover the entire plane, instead of leaving large empty areas (such as in Figure 16b). Since most of the plane is covered by polygons, little additional space is left for the class without a polygon, which reduces the risk of producing false positives.



(a) bcw (b) Iris

Fig. 16: Polygons on sample planes from the bcw dataset for the classes gray (no-recurrent) and white (recurrent), and the Iris dataset for the classes gray (setona), white (versicolour), black (virginica)

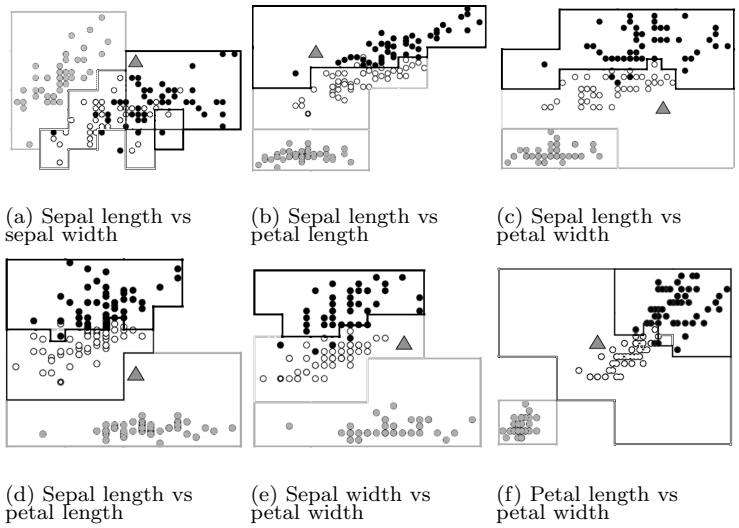
The number of polygons constructed is a trade-off between precision over speed. Adding one more polygon per plane increases the total training time, but the gain in classification accuracy can be significant. The focus of this paper is more on classification accuracy than on speed. We adopted the all-

class approach in this article, because the results in Table 4 demonstrate that this method can provide improved classification accuracy, and the increase in training time is not very large.

4.3.2 Proof of concept for many dimensions

In this section, we demonstrate the training phase and classification phase in PolyACO+ using the Iris dataset as an example (see section 4.1). Iris has 3 classes and 4 features: sepal width, sepal length, petal width and petal length. Iris has 4 dimensions, which produces 6 two-dimensional planes.

Figure 17 illustrates the dataset in each of the 6 planes. The surrounding polygons are produced during the training phase of PolyACO+. The color of each polygon corresponds to the class to which it belongs.



(a) Sepal length vs sepal width

(b) Sepal length vs petal length

(c) Sepal length vs petal width

(d) Sepal length vs petal length

(e) Sepal width vs petal width

(f) Petal length vs petal width

Fig. 17: Polygons for the Iris dataset in all planes with respect to classes gray (setona), white (versicolour), black (virginica). The triangle is a potential new point to be classified.

The accuracy level in Table 3 is very close to all variants of SVM. With respect to the Iris Plant dataset the accuracy level for PolyACO+ is 95.2, compared to 97.2 for linear SVM and 99.7 for polynomial SVM. It is noteworthy that PolyACO+ reaches an accuracy level higher than AntMiner+ 94.5, and, not surprisingly, significantly higher than the two-dimensional PolyACO. This is a quite understandable result because using all four features produces higher accuracy than only using two.

Hence, assuming that SVM and AntMiner+ are able to classify the data well in an adept manner, it could be argued that the PolyACO+ algorithm does so as well.

4.4 GPU performance comparison

In order to measure the improvement of PolyACO+ by introducing GPU parallelization, we ran an experiment comparing PolyACO+ with and without this component.

	Iris	**bcw**	**s-circ,100**	**s-circ,1000**	**s-circ,10000**
PolyACO+	113 sec	162 sec	6 sec	7 sec	8 sec
PolyACO+ without Multi-leveling	488 sec	698 sec	9 sec	10 sec	11 sec
PolyACO+ without GPU	685 sec	1956 sec	15 sec	77 sec	676 sec

Table 5: PolyACO+ feature by feature comparison using 10-fold cross-validation, measured in seconds used for training for Iris, bcw, semi-circular (s-circ) from 100 to 10,000 items.

The results in Table 5 show the runtime of PolyACO+ both with and without GPU parallelisation. The classification results themselves have been omitted from the table; as expected, they were very similar across all configurations. Applying parallelisation only changes the speed of the algorithm and not the classification accuracy. The gain in speed when using GPU is more significant when the size of the dataset increases.

Parallelisation reduces the runtime by a factor of up to 61.5x, which is a significant improvement compared with the original PolyACO (Tufteland et al, 2016).

4.5 PolyACO+ compared to other classification algorithms

In order to test how well PolyACO+ performs on an overall basis, we compare it to state-of-the-art algorithms. Thus, in this experiment we compare PolyACO+ to logistic regression, SVM, neural networks, AntMiner+ and the original PolyACO. These algorithms were chosen either because they are very popular in the machine learning community or have with similarities with PolyACO+. We used SVM and logistic regression implementations from the scikit-learn library and used Weka to run a neural network containing 2 hidden layers and a learning rate of 0.5. The AntMiner+ results are from the original AntMiner+ paper (Martens et al, 2007). The AntMiner+ results are

run with 10-fold cross-validation, while all other results are run with 100-fold cross-validation.[6]

4.5.1 Performance in higher-order dimensions

The training time of PolyACO+ drastically increases by increasing the number of dimensions due to the increase of the number of two-dimensional planes. For example, in the case of 4 dimensions, the number of planes is only $\binom{4}{2} = 6$, while for 1000 dimensions the number of planes is $\binom{1000}{2} = 499500$. On Iris, PolyACO+ uses on average 17.67 seconds per plane to achieve good accuracy (95% ± 2%). Regarding a dataset with 1,000 dimensions and equally many samples and classes as Iris, it would take approximately 100 days to complete a training phase.

In order for PolyACO+ to be practical to use in higher-order dimensions, some measures should be taken to either reduce the number of dimensions, e.g., through dimension reductions such as Principal Component Analysis (PCA) and t-Distributed Stochastic Neighbor Embedding (t-SNE) (Van Der Maaten, 2014) or apply other data reduction methods (Salama and Abdelbar, 2016; Varma et al, 2015), or to drastically increase the speed of the algorithm.

4.5.2 Results

Tables 3, 6, and 7 show that the classification scores are very close to each other, with the exception of PolyACO. PolyACO+ scores the highest on bcw, while SVM obtains the best score on Iris and neural networks scores best on Digits. The original PolyACO has the lowest score on all the datasets. This is probably because PolyACO can only classify two-dimensional data, and therefore it extracts only the first two features of each dataset and trains the model only based on them. It has a score of 10.18% on Digits, which is statistically equivalent to guessing as Digits has 10 classes. This is expected, since PolyACO only uses 2 out of the 64 features from Digits. Section 4.5.3 elaborates on PolyACO+ speed end convergence rate, including steps towards reaching an accuracy of 72.56% on the Digits data.

It should be noted that the best SVM setup from the experiments in Table 3 is, not surprisingly, Gaussian SVM. It is for this reason that we have included comparisons to Gaussian SVM in Table 6.

Table 6 presents comparison results of PolyACO+ with other state-of-the-art algorithms on real datasets. For the purpose of comparison, we have included all obtainable data from (Martens et al, 2007).[7] PolyACO+ produces a better accuracy level than all other algorithms for the three datasets Iris,

[6] The AntMiner+ data is from the original AntMiner+ paper (Martens et al, 2007) where experiments are run with 10-fold cross validation. A higher cross-validation produces less bias towards overestimating the true expected error.

[7] Two datasets from (Martens et al, 2007) could not be obtained because they are only available per request. Despite requesting it from the data provider, we unfortunately did not receive the data.

	aus	ttt	cmc	tae	bal	car	wine	iris	bcw	dig
Instances	690	958	1473	151	625	1728	178	151	699	1797
Attributes	15	9	9	5	4	6	13	4	9	64
Polygons	182	72	108	30	18	60	234	18	72	20160
num/cat	both	cat	both	cat	num	cat	num	num	num	num
Technique										
PolyACO+	**85.52**	65.80	45.05	49.70	87.43	69.91	**99.32**	**95.20**	**98.05**	78.28
AntMiner+	84.05	99.75	45.93	**56.73**	79.81	92.01	94.59	56.73	96.40	-
AntMiner	84.09	77.03	42.32	40.39	68.09	77.38	84.50	76.60	91.14	-
AntMiner2	84.30	71.13	41.49	43.73	66.94	77.93	85.33	81.80	91.54	-
AntMiner3	83.61	68.94	40.85	40.39	65.02	77.50	83.50	77.00	90.91	-
RIPPER	84.52	97.99	**48.94**	35.30	79.38	94.01	90.68	93.00	95.35	-
C4.5	84.82	83.79	46.60	47.20	77.11	96.61	89.83	93.80	94.69	-
1NN	80.33	98.50	42.16	50.20	81.83	92.69	95.43	91.00	96.40	-
logit	84.03	65.57	47.52	51.96	86.75	80.52	94.33	93.80	**96.53**	96.40
Gaussian SVM	85.22	**91.06**	48.55	48.42	**91.58**	**97.71**	94.83	94.40	92.81	94.01

Table 6: Classification accuracy of PolyACO+ compared to state-of-the-art classification algorithms for the datasets Australian Credit Approval (aus),Tic-Tac-Toe Endgame (ttt), Contraceptive Method Choice (cmc), Teaching Assistant Evaluation (tae), Balance Scale (bal), Car Evaluation (car), Wine, Iris, Breast Cancer Wisconsin (bcw), and Digits (dig). All data is from the UCI dataset repository (Lichman, 2013). Results other than PolyACO+ are from the original AntMiner+ paper (Martens et al, 2007)

bcw, wine, and better than all except Gaussian SVM for the bal dataset. However, more importantly, it is only outperformed by AntMiner+ with a large margin for three datasets (ttt, tae and car), all of which have categorical features. For all other results, it reaches roughly the same, or significantly higher, accuracy level. The natural conclusion to be drawn is that AntMiner+ is superior to PolyACO+ for categorical data, and the opposite seems to be the case for datasets with continues numerical values.

Table 7 shows accuracy categorised by data type (data containing categorical values, numerical values and both).

The confusion matrices in Figure 18 illustrate the precision optained when classifying Iris using PolyACO+. When comparing this matrix to the other classifiers' confusion matrix, we find that they all look approximately the same, which arguably supports the assertion that PolyACO+ works as well as comparable algorithms. Overall, the results from tables 3, 6 and the confusion matrices, strongly indicate that PolyACO+ is a competitive algorithm for solving classification problems when the features have numerical values. It is significantly more accurate than PolyACO, and performs similarly or better compared to the other algorithms in the experiment. However, it is less competitive when it comes to categorical data.

4.5.3 Performance versus convergence rate

Figure 19 shows how the classification accuracy of PolyACO+ stabilises at a higher convergence rate on the datasets Iris, bcw and Digits. The figure shows that while a higher convergence rate produces a higher classification accuracy, this accuracy is reached at different levels for different datasets. This is in line

Technique	cat	num	both	avg
PolyACO+	61.80	**95.00**	65.29	77.33
AntMiner+	**82.83**	81.88	64.99	78.44
AntMiner	64.93	80.08	63.21	71.28
AntMiner2	64.26	81.40	62.90	71.58
AntMiner3	62.28	79.10	62.23	69.75
RIPPER	75.67	89.60	66.73	79.90
C4.5	75.86	88.86	65.71	79.83
1NN	80.46	91.17	61.25	80.95
logit	66.01	92.85	65.78	77.89
Gaussian SVM	79.06	93.41	**66.89**	**82.73**

Table 7: Average classification accuracy of PolyACO+ compared to state-of-the-art classification algorithms for: datasets that only contain categorical values (cat), datasets that only contain numerical values (num), datasets that contain both categorical and numerical values (both), and average classification accuracy for all datasets (avg). Note that only PolyACO+, logit and SVM have been tested on Digits. Therefore, Digits is not included in the comparison.

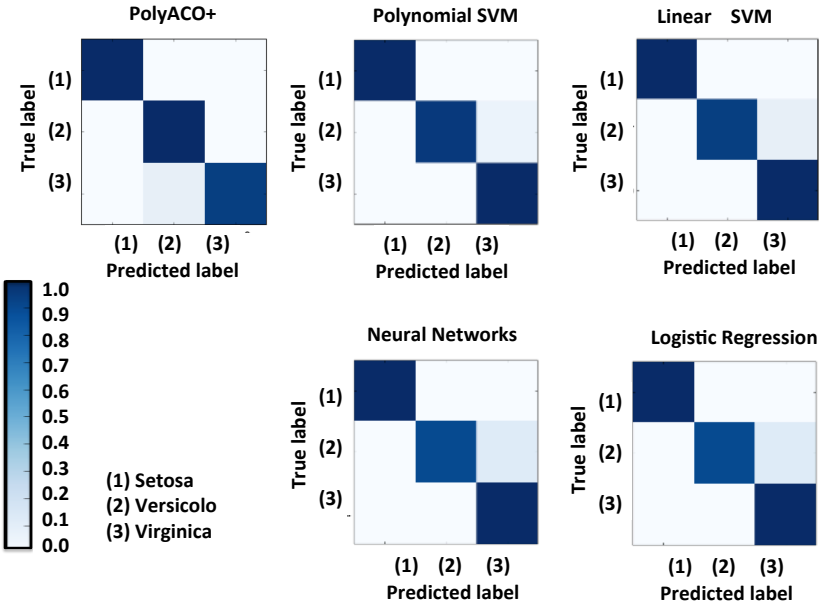


Fig. 18: Confusion matrices for different classifiers. Each is run once on Iris. PolyACO+, polynomial and linear SVM reached accuracies of 98%, while the other two scored 96% in this example.

with what might be expected as the datasets are of different complexity. Iris and bcw, which have 18 and 72 polygons respectively to optimise, converge for

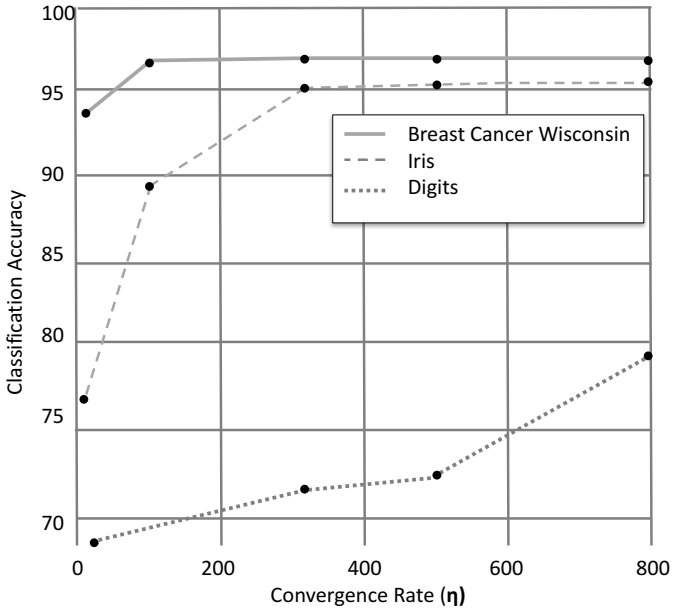


Fig. 19: Classification accuracy when using an increasing convergence rate (η) for PolyACO+ on Iris and bcw (average of 50 runs) and Digits (average of 5 runs)

$\eta \approx 300$. In order to train PolyACO+ for the more complex dataset Digits, it is needed to include 20160 polygons in the training phase, and the convergence rate clearly needs to be set higher. We would expect that by using a higher convergence rate, PolyACO+ would produce a higher level of accuracy, and it is therefore encouraging that the empirical evidence in Figure 19 supports this assertion.

4.6 Multi-level

This section compares the different multi-level approaches empirically. Each approach is run on the semi-circular dataset from Figure 12. The semi-circular dataset is a good choice to illustrate the effects of multi-leveling as there are clear boundaries between the class clusters in this case. The training phase is run for 100 seconds, and the best polygon is logged every second to monitor the progress. The four approaches in the experiment are: A static grid with granularity $\mu = 15$, naïve multi-level, static grid with AMR, and dynamic grid with AMR.

A value of $\mu = 15$ was empirically tested to give PolyACO+ the highest accuracy without multi-level (see Figure 8) and is included in Figure 20. The

results in Figure 20 show that the naïve multi-level approach converges earlier than the static grid approach with a fixed granularity, and at the same time lower than the two versions using AMR. Similarly, static AMR converges earlier than dynamic AMR, but with a lower score. This is because the static grid has a max level of 4, meaning that it has a small and limited search space. In this way, it not only converges very fast, but is not able to separate the data as well as the dynamic approach.

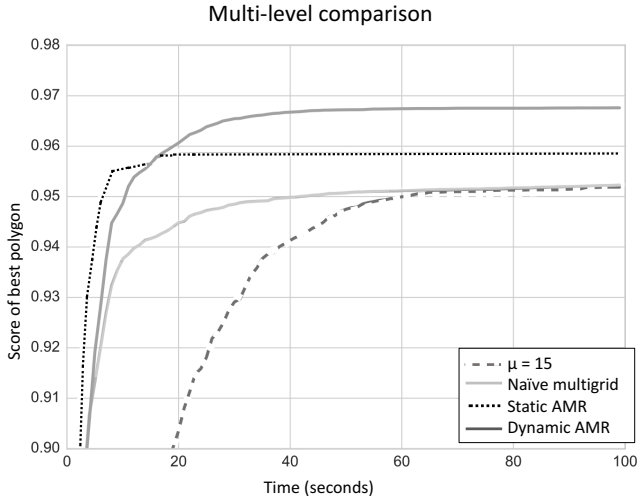


Fig. 20: Multi-level comparison (averaged over 50 runs)

4.6.1 Discussion

The naïve approach consists of applying traditional multigrid techniques by increasing the granularity on the entire grid when ants converge and pass on the pheromone values from the coarse grids to the finer grids. Another approach is to apply AMR by only increasing the granularity in areas of relevance. This approach can be applied either once at initialisation of the grid or dynamically during training. All multi-level approaches obtain very good results compared to no multi-leveling. Dynamic multi-leveling with AMR shows superior results compared to the other approaches and is therefore the preferred technique for PolyACO+. Moreover, the performance is further increased by applying a dynamic convergence which assigns more computing time to productive grid levels than to unproductive grid levels.

Convergence Rate	Average Accuracy	Average Time per run	Time per polygon
$\eta = 300$	72.55	69.6 hours	12 seconds
$\eta = 500$	72.89	112.4 hours	20 seconds
$\eta = 800$	78.28	139.8 hours	24 seconds

Table 8: Performance overview for Digits (average of 5 runs)

4.7 Performance on the digits dataset

Table 6 shows that PolyACO+ scores considerably lower on the Digits dataset than the other classification algorithms. This could be explained by the fact that the Digits dataset is more complex than the others, as it contains 64 features and 10 classes, resulting in a total of 20,160 polygons in the classification model (see Equation 7).

Figure 19 and Table 8 confirm that increasing the convergence rate also increases the accuracy level. However, this takes place at the expense of slower convergence since a higher convergence rate means increasing the number of ants.

4.8 Discrete, continues and categorical values

Table 7 shows the results for each classifier grouped by data type. When the data is purely numerical, PolyACO+ beats all algorithms in our comparison. In contrast, when the data is only categorical, AntMiner+ is the best performing algorithm. When the data contains both numerical and categorical values, SVM becomes the best performing algorithm but it is noteworthy that PolyACO+ is better than AntMiner+ in this case. One conclusion to be drawn from this is that PolyACO+ works best, and is the best among the compared algorithms, when the dataset contains numerical values. Without numerical values, PolyACO+ should not be the chosen classifier.

Figure 19 shows that PolyACO+ stabilises at a lower convergence rate on the bcw dataset than on Iris and Digits to stabilise. PolyACO+ achieves an average of 93.5% accuracy on bcw with $\eta = 10$, while it only achieves a 76.6% accuracy on Iris with the same settings.[8] The main difference between Iris and bcw is the number of features (4 and 9 respectively) and the type of values in the data: bcw only has discrete values between 1 and 10, while Iris has continuous values with no particular constraints. The multigrid in PolyACO+ is a discretisation of the entire dataset, and is therefore quite congruent with discrete values. Figure 21 illustrates the difference between multigrids based on continuous data and on discrete data. The grid can more easily separate the discrete data as it does not require a high resolution before each unique data position has a separate section in the grid. The continuous data requires a very high resolution to separate the points, because the values can be much closer to each other than in a discrete domain.

[8] Note that after convergence, the accuracy is much higher as presented in Table 6

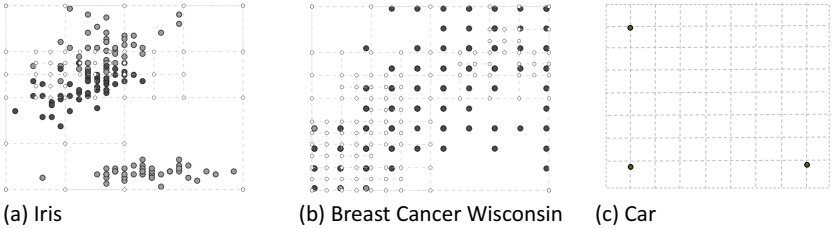


Fig. 21: Example of a multigrid in (a) continuous domain, Iris and (b) discrete domain, bcw, and (c) categorical data, Car

In order to explain the limitations of categorical data for PolyACO+, we use the example of the categorical dataset car. Applying PolyACO+ to the car dataset produces an accuracy level of 69.9, compared to 92.0 for AntMiner+.[9] The car dataset consists of 6 features, all of which are categorical. PolyACO+ by design expects numerical values for the features in a grid and not categorical features. We counter this problem by mapping every possible value in a category to a binary 0-1 category. For example, feature maintenance in the car dataset has 4 possible values: vhigh, high, med, and low. This is instead mapped to 4 features: maintenance_vhigh, maintenance_high, maintenance_med, and maintenance_low. Figure 21c shows a plot of maintenance_vhigh (x-axis) versus maintenance_high (y-axis). It is observable from this example that each items falls into either maintenance_vhigh, maintenance_high, or neither, and there is an exact overlap between these categories. Since PolyACO+ works in two-dimensional plots alone, there is not a very good separator from these features alone. Further, since the car dataset only includes categorical data, this is true for any subset of two features, and PolyACO+ does not perform that well in this situation.

5 Conclusion and further work

In this paper we have introduced PolyACO+, a classification algorithm which is a significant extension of the PolyACO algorithm that uses Ant Colony Optimisation and ray casting for classification. PolyACO+ introduces several modifications and improvements to PolyACO, such as the the capability to handle multi-dimensional data, support for multiple classes and an efficient parallel architecture for GPUs that improves accuracy and speed.

PolyACO+ employs a novel approach to handle multi-dimensional data where a separate classifier is trained individually for each pair of features. Subsequently, PolyACO+ uses ray casting and majority voting to construct

[9] Note that for the continues datasets, e.g. Iris, we observe the opposite case and PolyACO+ outperforms AntMiner+ by a large margin: 95.2 to 56.7.

an aggregated classification decision. The approach is successfully applied to multiple multidimensional datasets such as the bcw and Iris datasets. The algorithm performs similarly or better than state-of-the-art algorithms with an average classification accuracy of 95.20% on Iris and 97.11% on bcw. Moreover, PolyACO+ outperforms all other classification algorithms, including neural networks and SVM. Since bcw consists of discrete data, this suggests that PolyACO+ works particularly well for this type of data.

To reduce the complexity per dimension, PolyACO+ also applies adaptive mesh refinement. This multi-level technique works by increasing the granularity in parts of the graph with denser data while keeping lower granularity in parts where the data is sparse. In this manner, the ants do not have to spend time in areas which do not give added value.

Overall, PolyACO+ performs well on all our experiments with continuous and discrete numerical values, and meets our objectives in terms of improving PolyACO. We therefore conclude that PolyACO+ is a viable technique for solving classification problems. Concerning categorical data, PolyACO+ falls short compared to other classifiers specialised for this type of data, for example AntMiner+.

One of the biggest drawbacks of PolyACO is the fact that it is slow. In order to circumvent this disadvantage, PolyACO+ employs a parallel architecture for the reward function that can run on GPUs. The runtime of the reward function is reduced using paralellisation, which reduces the overall running time of the training phase to between 16% to 1.6% of its original runtime.

PolyACO+ is still relatively slow compared to other classical algorithms. The reason for this is that the number of polygons increases quadratically with respect to the number of features.

Two strategies might improve the performance further. First, extending parallelisation to other parts of the algorithm. Secondly, resorting to smart dimension reduction techniques. Potential approaches include adding a kernel functionality similar to how SVM uses kernels (i.e., smart exploration of relevant and non-relevant dimensions), and letting the ants walk in multidimensional planes. The viability of these approaches needs to be explored.

The paths in PolyACO+ are composed strictly of horizontal and vertical edges, and the distance along a path is computed using the Manhattan distance. Two paths that are of the same length calculated with the Manhattan distance, may be of different lengths in Euclidean geometry. The combination of PolyACO+ and a Euclidean geometry would permit diagonal lines and allow a greater variety of polygons. Whether or not Euclidean based PolyACO+ would improve the algorithm accuracy needs to be explored.

There are multiple ways of countering the Manhattan distance problem. One solution is to use the area of the polygon instead of the perimeter in the reward function. However, this approach could pose a challenge, as calculating the area may be computationally expensive, while calculating the polygon perimeter is trivial. Another approach would be to to introduce diagonal edges. This solves the problem of Manhattan distance, but comes at the cost of making the computations on the graph more expensive. Moreover it could be

challenging to implement the latter solution for asymmetric grids including the Adaptive Mesh Refinement grid used in PolyACO+.

Other mechanisms we intend to explore are multiple polygons per class per pair of dimensions and weighing of each polygon as both mechanisms would allow the detection of more complex patterns and potentially yield a higher level of classification accuracy. On the other hand, the mechanisms might as well increase the complexity of the classification phase. The weighing approach allows for more focused classification schemes when items are part of multiple classes. The weighing could be based on training accuracy or the distance an item is from the boundary.

Acknowledgement

The authors would like to thank the editors and anonymous referees for their unusually meticulous review and valuable comments to improve the quality of this paper.

References

Abadeh MS, Habibi J, Soroush E (2008) Induction of fuzzy classification systems via evolutionary ACO-based algorithms. First Asia International Conference on Modelling & Simulation, 2007. AMS'07. (pp. 346-351). IEEE.

Albinati J, Oliveira SE, Otero FE, Pappa GL (2015) An ant colony-based semi-supervised approach for learning classification rules. Swarm Intelligence 9(4):315–341.

Aribarg T, Supratid S, Lursinsap C (2012) Optimizing the modified fuzzy ant-miner for efficient medical diagnosis. Applied Intelligence 37(3):357–376.

Berger MJ, Colella P (1989) Local adaptive mesh refinement for shock hydrodynamics. Journal of Computational Physics 82(1):64–84.

Brandt A (1977) Multi-level adaptive solutions to boundary-value problems. Mathematics of computation 31(138):333–390.

Brandt A (1988) Multilevel computations: Review and recent developments. In: S. F. McCormik (Ed), Multigrid methods: Theory, applications, and supercomputing, proceedings of the 3rd Copper Mountain conference on multigrid methods. Lecture Notes in Pure and Appl Math, (vol 110, pp. 35–62).

Buluc A, Meyerhenke H, Safro I, Sanders P, Schulz C (2013) Recent advances in graph partitioning. In: L. Kliemann and P. Sanders (Eds), Algorithm Engineering (pp. 117-158) Springer.

Caruana R, Karampatziakis N, Yessenalina A (2008) An empirical evaluation of supervised learning in high dimensions. In: W. Cohen, A. McCallum & S. Roweis (Eds), Proceedings of the 25th international conference on Machine learning (pp. 96–193). ACM.

Daly R, Shen Q, et al (2009) Learning Bayesian network equivalence classes with Ant Colony Optimization. Journal of Artificial Intelligence Research 35(1):391.

Daly R, Shen Q, Aitken S (2011) Learning Bayesian networks: approaches and issues. The Knowledge Engineering Review 26(02):99–157.

De Campos LM, Puerta J, et al (2008) Learning Bayesian networks by ant colony optimisation: Searching in two different spaces. Mathware & Soft Computing 9(3):251–268.

Garey MR, Johnson DS (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness, 1st edn. W. H. Freeman, San Francisco.

Goodwin M, Yazidi A (2016) Ant colony optimisation-based classification using two-dimensional polygons. In: M. Dorigo, M. Birattari, X. Li, M. Lòpez-Ibáñez, K. Ohkura, C. Pinciroli & T. Stüzle (Eds.), International Conference on Swarm Intelligence (Proceedings ANTS-2016), Lecture Notes in Computers Science (Vol. 9882, pp. 53–64). Springer.

Goodwin M, Yazidi A, Møller T (2016) Distributed learning automata for solving a classification task. In: 2016 IEEE Congress on Evolutionary Computation (CEC) (pp. 3999–4006). IEEE.

Jun-Zhong J, Zhang HX, Ren-Bing H, Chun-Nian L (2009) A Bayesian network learning algorithm based on independence test and ant colony optimization. Acta Automatica Sinica 35(3):281–288.

Klein RI (1999) Star formation with 3-D adaptive mesh refinement: The collapse and fragmentation of molecular clouds. Journal of Computational and Applied Mathematics 109(1):123–152.

Lian TA, Llave MR, Goodwin M, Bouhmala N (2015) Towards multilevel ant colony optimisation for the Euclidean symmetric traveling salesman problem. In: Ali M., Kwon Y., Lee CH., Kim J., Kim Y. (Eds.), Current Approaches in Applied Artificial Intelligence (Proceedings IEA/AIE 2015), Lecture Notes in Computer Science (Vol 9101, pp. 222–231). Springer.

Lichman M (2013) UCI machine learning repository. URL `http://archive.ics.uci.edu/ml`.

Liu B, Abbas H, McKay B (2003) Classification rule discovery with Ant Colony Optimization. In: IEEE/WIC International Conference on Intelligent Agent Technology, 2003. IAT 2003 (pp. 83–88). IEEE.

Madjarov G, Kocev D, Gjorgjevikj D, Džeroski S (2012) An extensive experimental comparison of methods for multi-label learning. Pattern Recognition 45(9):3084–3104.

Martens D, Backer MD, Haesen R, Vanthienen J, Snoeck M, Baesens B (2007) Classification with ant colony optimization. IEEE Transactions on Evolutionary Computation 11(5):651–665.

Martens D, Baesens B, Fawcett T (2011) Editorial survey: Swarm intelligence for data mining. Machine Learning 82(1):1–42.

Parpinelli R, Lopes H, Freitas A (2002) Data mining with an ant colony optimization algorithm. IEEE Transactions on Evolutionary Computation 6(4):321–332.

Ryoo S, Rodrigues CI, Baghsorkhi SS, Stone SS, Kirk DB, Hwu WmW (2008) Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. In: Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (pp. 73-82). ACM.

Salama KM, Abdelbar AM (2015) Learning neural network structures with ant colony algorithms. Swarm Intelligence 9(4):229–265.

Salama KM, Abdelbar AM (2016) Using Ant Colony Optimization to build cluster-based classification systems. In: M. Dorigo, M. Birattari, X. Li, M. Lòpez-Ibáñez, K. Ohkura, C. Pinciroli & T. Stüzle (Eds.), International Conference on Swarm Intelligence (Proceedings ANTS-2016), Lecture Notes in Computers Science (Vol. 9882, pp. 210–222). Springer.

Sapin E, Keedwell E, Frayling T (2015) Ant colony optimisation of decision tree and contingency table models for the discovery of gene interactions. IET Systems Biology 9(6):218–225.

Sharma S., Ghosh S., Anantharaman N., Jayaraman V.K. (2012) Simultaneous informative gene extraction and cancer classification using ACO-AntMiner and ACO-Random forests. In: Satapathy S.C., Avadhani P.S., Abraham A. (Eds) Proceedings of the International Conference on Information Systems Design and Intelligent Applications 2012 (INDIA 2012) Advances in Intelligent and Soft Computing, (vol 132, pp. 755–761). Springer.

Smola AJ, Schölkopf B (2004) A tutorial on support vector regression. Statistics and Computing 14(3):199–222.

Stützle T, Hoos HH (2000) MAX–MIN Ant System. Future Generation Computer Systems 16(8):889–914.

Tripathy S, Hota S, Satapathy P (2013) MTACO-Miner: Modified threshold ant colony optimization miner for classification rule mining. In: N. R. Shetty, N. H. Prasad & N. Nalini (Eds), International Conference on Emerging Research in Computing, Information, Communication and Applicagtions. (pp. 529-534). Elsevier.

Tufteland T., desneltvedt G., Goodwin M. (2016) Optimizing PolyACO training with GPU-based parallelization. In: M. Dorigo, M. Birattari, X. Li, M. Lòpez-Ibáñez, K. Ohkura, C. Pinciroli & T. Stüzle (Eds.), International Conference on Swarm Intelligence (Proceedings ANTS-2016), Lecture Notes in Computers Science (Vol. 9882, pp. 233–240). Springer.

Van Der Maaten L (2014) Accelerating t-SNE using tree-based algorithms. Journal of Machine Learning research 15(1):3221–3245.

Varma PRK, Kumari VV, Kumar SS (2015) A novel rough set attribute reduction based on ant colony optimisation. International Journal of Intelligent Systems Technologies and Applications 14(3-4):330–353.

Walshaw C (2004) Multilevel Refinement for Combinatorial Optimisation Problems. Annals of Operations Research 131(1-4):325–372.

Xue B, Zhang M, Browne WN (2014) Particle swarm optimisation for feature selection in classification: Novel initialisation and updating mechanisms. Applied Soft Computing 18:261–276.