A Hybrid PSO/ACO Algorithm for Classification

Nicholas Holden
University of Kent
Computing Laboratory
Canterbury, CT2 7NF, UK
+44 (0)1227 823192
nickpholden@gmail.com

Alex A. Frietas
University of Kent
Computing Laboratory
Canterbury, CT2 7NF, UK
+44 (0)1227 827220
A.A.Freitas@kent.ac.uk

ABSTRACT
In a previous work we have proposed a hybrid Particle Swarm Optimisation/Ant Colony Optimisation (PSO/ACO) algorithm for the discovery of classification rules, in the context of data mining. Unlike a conventional PSO algorithm, this hybrid algorithm can directly cope with nominal attributes, without converting nominal values into numbers in a pre-processing phase. The design of this hybrid algorithm was motivated by the fact that nominal attributes are common in data mining, but the algorithm can in principle be applied to other kinds of problems involving nominal variables (though this paper focuses only on data mining). In this paper we propose several modifications to the original PSO/ACO algorithm. We evaluate the new version of the PSO/ACO algorithm (PSO/ACO2) in 16 public-domain real-world datasets often used to benchmark the performance of classification algorithms. PSO/ACO2 is evaluated with two different rule quality (particle "fitness") functions. We show that the choice of rule quality measure greatly effects the end performance of PSO/ACO2. In addition, the results show that PSO/ACO2 is very competitive with respect to two well-known rule induction algorithms.

Categories and Subject Descriptors
I.2.6 **[Artificial Intelligence]**: Learning – *Induction.*

General Terms: Algorithms.

Keywords: Rule Induction, Classification, Particle Swarm optimization, Ant Colony Optimisation, Data Mining.

1. INTRODUCTION
We have previously proposed a hybrid Particle Swarm Optimisation/Ant Colony Optimisation (PSO/ACO) algorithm for the discovery of classification rules, in the context of data mining. (A brief review of classification rules is provided in Section 2.) Unlike a conventional PSO algorithm, this hybrid algorithm can directly cope with not only continuous but also nominal attributes, without converting nominal values into numbers in a pre-processing phase. The design of this hybrid algorithm was motivated by the fact that nominal attributes are common in data mining, but the algorithm can in principle be applied to other kinds of problems involving nominal variables (though this paper focuses only on data mining).

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

In this paper we propose several modifications to the original PSO/ACO algorithm. In essence, the proposed modifications involve changes in the pheromone updating procedure and in the rule initialization method, as well as – significantly – the splitting of the rule discovery process into two separate phases, as follows. In the first phase a rule is discovered using nominal attributes only. In the second phase the rule is potentially extended with continuous attributes. This further increases the ability of the PSO/ACO algorithm in treating nominal and continuous attributes in different ways, recognizing the differences in these two kinds of attributes (a fact ignored by a conventional PSO algorithm, as mentioned earlier). We also experiment with two different rule quality functions, since the choice of a such a function is a very important aspect in the design of a data mining algorithm. We evaluate the new version of the PSO/ACO algorithm – denoted PSO/ACO2 in this paper – in 16 public-domain real-world datasets often used to benchmark the performance of classification algorithms.

The remainder of the paper is organised as follows. Section 2 introduces the classification task, section 3 describes in detail the workings of the modified algorithm. Section 4 discusses the reasons for the modifications, in section 5 we present the experimental set-up and results. In section 6 we draw some conclusions from the work and discuss possible future research.

2. CLASSIFICATION
The task (kind of problem) addressed in this paper is the classification task of data mining. In classification the knowledge or patterns discovered in the data set can be represented in terms of a set of rules. A rule consists of an antecedent (a set of attribute-values) and a consequent (class):

IF <attrib = value> AND ... AND <attrib = value> THEN <class>

The consequent of the rule is the class that is predicted by that rule. The antecedent consists of a set of terms, where each term is essentially an attribute-value pair. More precisely, a term is defined by a triple <*attribute, operator, value*>, where *value* is a value belonging to the domain of *attribute*. The *operator* used in this paper is "=" in the case of categorical/nominal attributes, or "≤" and ">" in the case of continuous attributes. The knowledge representation in the form of rules has the advantage of being intuitively comprehensible to the user. This is important, because the general goal of data mining is to discover knowledge that is not only accurate, but also comprehensible [12][3].

3. THE MODIFIED PSO/ACO ALGORITHM
In this section we first provide a very brief overview of the hybrid Particle Swarm Optimization/Ant Colony Optimization (PSO/ACO) algorithm originally proposed in [5] and [6] – hereafter denoted as PSO/ACO1. This algorithm was designed to be the first PSO-based

classification algorithm to natively support nominal data – i.e., to cope with nominal data directly, without converting a nominal attribute into a numeric or binary one and then applying a mathematical operator to the value. The motivation to natively support nominal data is that by converting a nominal attribute such as *gender* into a numerical attribute (say, mapping *male* into 0 and *female* into 1) we would be introducing an artificial order among the numerical values (1 > 0). Such an order clearly makes no sense in the context of original nominal values, and mathematical operations applied to this artificial order may generate counter-intuitive results.

The PSO/ACO1 algorithm achieves a native support of nominal data by combining ideas from Ant Colony Optimisation (Ant-Miner classification algorithm [10]) and Particle Swarm Optimisation [8][11] to create a classification meta heuristic that supports innately both nominal (including binary as a special case) and continuous attribute.

3.1.PSO/ACO2 Sequential Covering Approach

RS = ∅ /* initially, Rule Set is empty */
FOR EACH class *C*
    TS = {All training examples}
    WHILE (Number of uncovered training examples
        belonging to class *C* > *MaxUncovExampPerClass*)
        Run the PSO/ACO algorithm to discover the
        best nominal rule predicting class *C*, called *Rule*
        Run the standard PSO algorithm to add continuous terms
        to *Rule,* and return the best discovered rule *BestRule*
        Prune *BestRule*
        RS = *RS* ∪ *BestRule*
        TS = *TS* – {training examples correctly covered by
            discovered rule}
    END WHILE
END FOR
Order rules in *RS* by decending Quality

Pseudocode 1: Sequential Covering Approach used by the Hybrid PSO/ACO2 Algorithm

Both the original PSO/ACO algorithm and the new modified version uses a sequential covering approach [12] to discover one-classification-rule-at-a-time. The original PSO/ACO algorithm is described in detail in [5][6], hereafter we describe how the sequential covering approach is used in the new modified version of the PSO/ACO algorithm as described in Pseudocode 1 – hereafter denoted as PSO/ACO2. It starts by initialising the rule set (*RS*) with the empty set. Then, for each class the algorithm performs a WHILE loop, *TS* is used to store the set of training examples the rules will be created from. Each iteration of this loop performs one run of the PSO/ACO2 algorithm which only discovers rules based on nominal attributes, returning the best discovered rule *(Rule)* predicting examples of the current class (C). The rule returned by the PSO/ACO2 algorithm is not (usually) complete as it does not include any terms with continuous values. For this to happen the best rule discovered by the PSO/ACO2 algorithm is used as a base for the discovery of terms with continuous values. A standard PSO algorithm (applied only to numeric attributes) is used [1] with constriction. The vector to be optimised consists of two dimensions per continuous attribute, one for an upper bound and one for a lower bound. At every particle evaluation the vector is converted to set of terms (rule conditions) and added to *Rule* produced by the PSO/ACO2 algorithm for fitness evaluation. For instance, if the data set contained one nominal attribute A_{n0} and one continuous attribute A_{c0} the PSO/ACO algorithm might produce a rule like: IF A_{n0} = <*value*> THEN class *C*. The standard PSO algorithm would then attempt to improve this rule by adding terms: $x_{i1} > A_{c0}$ AND $x_{i2} \leq A_{c0}$, which effectively corresponds to a term of the form: $x_{i1} > A_{c0} \geq x_{i2}$. Where a particle's position would be the vector x_{i1}, x_{i2}. The rule for evaluation purposes would be:

IF A_{n0} = <*value*> AND $x_{i1} > A_{c0}$ AND $x_{i2} \leq A_{c0}$ THEN class *C*

If the two bounds cross over (i.e., $0 > A_{c0} \geq 1$) both terms are omitted from the decoded rule but the PBest (p_i) positions is still updated in those dimensions.

$$v_{id} = \chi\,(v_{id} + c_1\epsilon_1\,(p_{id} - x_{id}) + c_2\epsilon_2\,(p_{gd} - x_{id}))$$

Equation 1: PSO Velocity Update

$$x_{id} = x_{id} + v_{id}$$

Equation 2: PSO Position Update

The best rule is then added to the rule set after being pruned using Ant-Miner "style" pruning [10], and the examples correctly covered by that rule are removed from the training set (*TS*). An example is said to be correctly covered by a rule if that example satisfies all the terms (attribute-value pairs) in the rule antecedent ("IF part") and it has the class predicted by the rule. This WHILE loop is performed as long as the number of uncovered examples of the class *C* is greater than a user-defined threshold, the maximum number of uncovered examples per class (*MaxUncovExampPerClass*). After this threshold has been reached *TS* is restored to all the original training examples. This process means that the rule set generated is unordered – it is possible to use the rules in the rule set in any order to classify an example without unnecessary degradation of predictive accuracy. Having an unordered rule set is important because after the entire rule set is created the rules are ordered by their Quality and not the order they were created in, although this is a greedy approach it improved predictive accuracy in initial experiments.

3.2.The PSO/ACO2 Algorithm

Each particle in the PSO/ACO2 population is a collection of *n* pheromone matrices where *n* is the number of nominal attributes in a data set. Each particle can be decoded probabilistically into a rule with a predefined consequent class. Each of the *n* matrices has two entries, one entry represents an *off* state and one entry represents an *on* state. If the *off* state is selected the corresponding attribute-value pair will not be included in the decoded rule. If the *on* state is selected when the rule is decoded the corresponding attribute-value pair will be added to the decoded rule. Which value is included in this attribute-value pair (term) is dependant on the seeding values. The seeding values are set when the population of particles is initialised, each particle has its past best state (which is a rule) set to a randomly chosen record (a collection of attribute-value pairs) with class *C* – the same class as the predefined consequent class for the decoded rules. From now on the particle is only able to translate to a rule with attribute-values equal to the seeding terms, or to a rule without some or all those terms. This may seem at first glance, counter-intuitive as flexibility is lost – each particle cannot translate into any possible rule, the reasons for this will be discussed later. Each pheromone matrix entry is a real valued number and all the entries in each matrix for each attribute add up to 1. Each entry in each pheromone matrix is associated with a minimum, positive, non-zero pheromone value. This prevents a pheromone from dropping to zero, helping to increase the diversity of the population (reducing the risk of premature convergence).

```
Initialize population
REPEAT for MaxInterations
  FOR every particle P
   /* Rule Creation */
   Set Rule R = "IF ∅ THEN C"
   FOR ever dimension d in P
     Use roulette selection to choose whether the state should be set
     to off or on. If it is on then the corresponding attribute-value pair
     set in the initialisation will be added to R otherwise, if off is
     selected nothing will be added.
   LOOP
   Calculate Quality Q of R
   /* Set the past best position */
   IF Q > P's Best past rule's (R_p) Quality Q_p
    Q_p= Q
    R_p = R
   END IF
  LOOP
  FOR every particle P
   Find best Neighbour Particle N according to N's Q_p
   FOR every dimension d in P
    /* Pheromone updating procedure */
    IF best state selected for P_d = best state selected for N_d THEN
     pheromone_entry for the best state selected for P_d is
     increased by Q_p
    ELSE
     pheromone_entry for the best state selected for P_d is
     decreased by Q
    END IF
    Normalize pheromone_entries
   LOOP
  LOOP
LOOP
```

Pseudocode 2: The Hybrid PSO/ACO Algorithm

Pseudocode 2 shows the modified PSO/ACO algorithm proposed in this paper and utilised in Pseudocode 1. Firstly the population of particles is initialised. Each particle is seeded with terms from a randomly selected record, as described in the previous paragraph. Initially, in each dimension the pheromone for the *on* state is set to ≈ 0.9 and the pheromone for the *off* state is set to ≈ 0.1. The first loop iterates the whole population for *MaxIterations*. Then for each particle P a rule is built probabilistically from its pheromone matrices. For each dimension d in P, roulette selection is used to decide if the *on* or *off* state should be selected [5]. If the *on* state is selected then the corresponding term is added to the antecedent of R, this is an attribute-value pair where the attribute corresponds to the dimension d and the value corresponds to the initial seeding value. After this process has been repeated for every dimension, the quality Q of the rule is calculated. If the new Q is greater than the previous best Q_p, then the particle's state (R) is saved as R_p.

After the rule creation phase the pheromone is updated for every particle. Each particle finds its best neighbour (N) according to the rule quality measure (as saved in Q_p). For every dimension d in P and so every corresponding dimension in N the following pheromone updating rules are used. If the best state selected for P in the dimension d (P_{Bestd}) is the same as the best state selected for N in d then an amount of pheromone equal to Q_p is added to the pheromone entry for the best state for P in d. If the best state selected for P in the dimension d is *not* the same as the best state selected for N in d then an amount of pheromone equal to Q is removed from the pheromone entry for the best state for P in d. If after this process is completed any pheromone entry is less than a minimum amount then it is set to that amount (≈ 0.001). Importantly this allows the pheromone entry that is not the best state to increase due to normalisation, it also aids search in a conveptually similar way to mutation in GAs. This is because if the non best state reached a pheromone value of zero then it would always remain at zero even if pheromone was being removed from the other entry. After the pheromone matrix has been updated it is normalised, so that its entries add up to one.

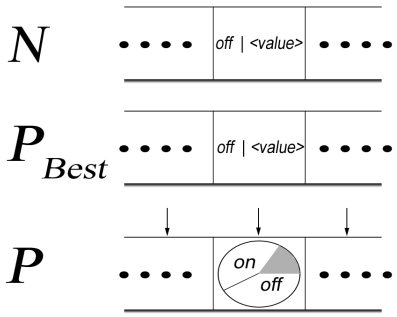

Figure 1: Pheromone Update in PSO/ACO2

Table 1: Different Pheromone Updating Scenarios

State for N_d	Sate for P_{Bestd}	Outcome for entries in P_d
(*on*) <*value*>=*X*	(*on*) <*value*>=*X*	*on* pheromone increased *off* pheromone decreased
(*on*) <*value*>=*X*	(*on*) <*value*>=*Y*	*off* pheromone increased *on* pheromone decreased
off	(*on*) <*value*>=*X*	*off* pheromone increased *on* pheromone decreased
off	*off*	*off* pheromone increased *on* pheromone decreased

Figure 1 gives a graphical representation of how both the personal best P_{Best} and the best Neighbour N effect the pheromone in the current particle P. In table 1 the four possible scenarios for pheromone updating are described given the differing states of P_{Best} and N. Note that the only time the pheromone for the entry *on* is increased (by Q_p) is when the state of P_{Best} and N are set to *on* and their corresponding values are the same. In all other cases the pheromone is decreased (by Q).

3.3.Quality Measures

It is necessary to estimate the quality of every candidate rule (decoded particle). A measure must be used in the training phase in an attempt to estimate how well a rule will perform in the testing phase. Given such a measure it becomes possible to optimise a rule's quality (the fitness function) in the training phase and this is the aim of the PSO/ACO2 algorithm. In our previous work [5] the Quality measure used was Sensitivity × Specificity (Equation 3) [4]. Where TP, FN, FP and TN are, respectively, the number of true positives, false negatives, false positives and true negatives associated with the rule [12].

$$Sensitivity \times Specificity = TP / (TP + FN) \times TN / (TN + FP)$$

Equation 3: Original Quality Measure

Later we found that Equation 3 does not give an effective measure of accuracy under certain circumstances [6]. This led us to modify the quality measure so that when the class predicted by a rule is the majority class (a class having more examples than the rest of the data set combined) its quality is computed by the product of the rule's sensitivity and specificity as shown in Equation 3. When a rule predicts a minority class (i.e., any class different from the majority class) the product of sensitivity and precision [6], shown in Equation 4, is used as the rule's quality.

$$Sensitivity \times Precision = TP / (TP + FN) \times TP / (TP + FP)$$

Equation 4: Quality Measure on Minority Class

Subsequent experiments proved that although this new measure of quality deals better with extreme cases it caused rules to be generated that produced suboptimal accuracies in the testing phase. We have modified the quality measure so that when the minority class is being predicted precision with Laplace correction [12] [2] is used, as per equation 5.

$$Precision = 1 + TP / (1 + k + TP + FP)$$

Equation 5: New Quality Measure on Minority Class

Where k is the number of classes. We noticed that in some cases the use of Precision as a rule quality measure would lead to rules covering very few examples. To stop this less than ideal situation we also added the following conditional statement to the new quality measure:

IF $TP < MinTP$
  $Quality = 0$
ELSE
  $Quality = Precision$
END IF

Where *MinTP* is the least number of correctly covered examples that a rule has to cover before it is given a positive non-zero quality. In our experiments we set *MinTP* to equal 10 but any comparably small number will have a similar effect. These two different quality measures are compared in section 5.

4. MOTIVATIONS, MODIFICATIONS AND DISCUSSION

The modified algorithm (PSO/ACO2) proposed in this paper differs from the original algorithm (PSO/ACO1) proposed in [5][6] in four important ways. Firstly PSO/ACO1 attempted to optimise both the continuous and nominal attributes present in a rule antecedent at the same time, whereas PSO/ACO2 takes the best nominal rule built by PSO/ACO2 and then attempts to add continuous attributes using a standard PSO algorithm. Secondly the original algorithm used a type of pruning to create seeding terms for each particle, PSO/ACO2 uses all the terms from an entire training example (record). Thirdly in PSO/ACO1 it was possible for a particle to select a value for an attribute that was not present in its seeding terms, in PSO/ACO2 only the seeding terms' values may be added to the decoded rule. Fourthly the pheromone updating rules have been simplified to concentrate on the optimisational properties of the original algorithm. In PSO/ACO1 pheromone was added to each entry that corresponded to the particle's past best state, its best neighbour's best state, and the particle's current state in proportion to a random learning factor. Now pheromone is only added when the best neighbour's best state and the current particle's best state match, or taken away when they do not. Also pheromone is only ever added to the best state's entry.

In PSO/ACO2 (Pseudocode 2) the standard PSO and the hybrid PSO/ACO2 algorithms have been separated partially because they differ quite largely in the time taken to reach peak fitness. It usually takes about 30 iterations for the pheromone matrices to reach a stable state in PSO/ACO2, it tends to take considerably longer for the standard PSO algorithm to converge. Due to this fact the standard PSO algorithm's particles set *PBests* early on in quite dissimilar positions, this causes high velocities and suboptimal search, with a higher likelihood of missing a position of high fitness. So separating the two algorithms provides more consistent results.

Secondly in the PSO/ACO1 algorithm, sets of seeding terms were pruned before they were used, the aggressive pruning algorithm used a heuristic to discard certain terms. This is less than ideal as the heuristic does not take into account attribute interaction, and so potentially useful terms are not investigated.

To understand the reasons behind the last two modifications it is important to understand how the algorithms find good rules. In both PSO/ACO1 and PSO/ACO2 sets of terms are generated by mixing together the experiences of the particles and their neighbours. This mixing is done on a performance driven basis, so a particle will only be influenced by its best neighbour, this means that areas of promising quality are investigated. Each particle is a probabilistic rule generator. As the pheromone entries in the matrices converge and reach one (and zero), better rules should be generated more often.

In PSO/ACO1 the levels of the pheromone in the matrices is influenced by three factors (current state, past best state and best neighbours' best state) [5]. If these factors do not agree then the pheromone matrix will be slow to converge. Slow convergence can be advantageous in this type of situation as the algorithm should not prematurely converge on a local maxima. However, in PSO/ACO1 the result of this slow convergence is usually destructive, as incompatible terms can be mixed together over and over again. Incompatible terms are terms that do not cover any of the same records. In Table 2, example, incompatible terms are A_{n1} = a and A_{n2} = b. A rule including both these terms would have a quality of zero as it would not cover any examples. This problem is addressed by the third modification in PSO/ACO2, now incompatible terms will not be mixed.

In PSO/ACO2 the pattern being investigated by the particles will likely include relatively general terms – an example might be a rule including the term A_{n3} = b in table 2. It is the job of the PSO/ACO2 algorithm to find terms that interact well together to create a rule that is not only general to the class being predicted (and so particles representing records, or parts of records from it) but also specific to the class as well (by not covering examples in other classes). It is also the job of the PSO/ACO2 algorithm to turn off terms that limit the generality of the rule without adding specificity to it. This trade-off between specificity and generality (or sensitivity) is calculated by the rule quality measure. It is clear, in Table 2, that including values for A_{n1} and A_{n2} will not create a good trade-off between sensitivity and specificity and so due to the new pheromone updating procedures a particle will choose the *off* state for these conflicting attributes. As this is a good modification it will propagate throughout the population. These are the most important qualities of the PSO/ACO2 algorithm and are exploited more effectively by the new pheromone updating procedure.

Modification three in PSO/ACO2 ensures a particle will always cover at least one example (the seeding example) even if all the

terms are included in the decoded rule. This was not the case in PSO/ACO1 as at the beginning of the search many incompatible terms would be mixed creating many particles with zero fitness .

Table 2. An Example Single Class Data Set, R's are Records, A_n's are Nominal Attributes

	A_{n1}	A_{n2}	A_{n3}
R_1	a	a	a
R_2	a	a	b
R_3	a	a	b
R_4	b	b	b
R_5	b	b	b
R_6	b	b	b

For instance, if record R_1 and R_4 are used to seed two different particles then three attribute value conflicts will take place, this is due to the fact that no single rule can cover both examples. In PSO/ACO1 the particle seeded with R_1 would most likely eventually be converted to a particle more suited to covering R_4 as there are more records within the data set that follow R_4's pattern. However, it is not clear that this is always a good idea because, in the PSO/ACO algorithm it is important to maintain diversity within the search for as long as possible. This is because the algorithm does not know which generalisations (arising from the interaction of particles starting with different seeding terms) will prove to be the best in the long term. In the example provided diversity is not important, but if R_1 characterised the pattern found within many other records then it would be, as this may lead to a rule with higher quality than the ones that may be generated after starting with R_4. Obviously too much diversity within the population would not be useful either, as a sort of hierarchy of interaction needs to take place to form good generalisations. The interaction between just two particles may not necessarily form a good rule, but the interaction between those two particles and another two may. In PSO/ACO2 more diversity is maintained because a particle seeded with R_1 can never be "converted" to cover R_4, it cannot produce a rule with a term $A_3 = b$. This may not initially seem like a good thing, as the population can never fully converge on a single solution. It is, however, not necessary for the whole population to converge to produce good rules with a reasonably sized and well connected population. This is because smaller "colonies" within the main population emerge that are compatible and mix to produce good and increasingly general rules for their shared pattern. Particles bordering the boundaries between different "colonies" will be unstable and have a low fitness but serve as a buffer to keep unhelpful influences out. If R_2 and R_4 were used to seed two particles then it is easy to see how the interactions of just these two particles would produce the optimal rule (with the single term $A_{n3} = b$) very quickly. It is the guided (according to a fitness function) generalising power of the algorithm, along with the ability to maintain diversity that also makes PSO/ACO2 effective and this is also reflected in the modified pheromone updating rules.

The pheromone updating rules (as can be seen in Table 1) simply say that, for a given dimension *d,* if the current particle *P* has the same term (attribute-value) in *d* as its best neighbour then reinforce the probability of selecting that term by adding pheromone, if they do not agree then increase the likelihood that the term will be turned *off* (via normalisation - evaporation). In this way it is possible to generate a very general (but specific to the current class and so of high quality) rule that most particles will have converged to by the end of the run. If there are only less general rules possible then many of them will be present throughout the population at the end, making it possible to select the best one.

5. RESULTS

For the experiments we used 16 of data sets from the well-known UCI dataset repository [9], we performed 10 fold cross validation, and run each algorithm 10 times for each fold for the stochastic algorithms.

Both the PSO/ACO2 algorithm and the standard PSO algorithm use the Von-Neumann topology as it is recommended as a good topology [7] and performed well in initial experiments. Both algorithms had 100 particles, the PSO/ACO2 algorithm ran for a maximum of 100 iterations (*MaxInterations*) and the standard PSO algorithm ran for a maximum of 100 iterations per rule discovered. In all experiments constriction factor $\chi = 0.72984$ and social and personal learning coefficients $c1 = c2 = 2.05$ [1]. *MaxUncovExampPerClass* = 2 as this is the minimum number of examples you can discover any sort of general rule for. The WEKA [12] statistics class was used to compute the standard deviation and two tailed Student T-Tests in the results presented.

The algorithms compared in table 3 are JRip [12] (WEKA's implementation of RIPPER) and PART [12] (which uses J48, WEKA's improved implementation of C4.5, to generate pruned decision trees from which rules are extracted). PSO/ACO2 (S×P) is the proposed variant of the algorithm that uses Sensitivity × Precision as the quality measure. PSO/ACO2 (P) is the variant of the proposed algorithm that uses precision with Laplace correction. We report results only for PSO/ACO2 as in initial experiments PSO/ACO2 always outperformed PSO/ACO1. The shading in the last three columns of the table denotes a win or a loss (according to the two tailed Student's T-Test), light grey for a win and dark grey for a loss against the baseline algorithm (JRIP). It can be seen that overall PART scores equally as well as JRip, but PSO/ACO (P) wins by 2 (i.e., 3 wins against 1 loss) according to the two tailed T-Test. PSO/ACO (S×P) performs significantly worse overall losing by 5. Therefore against the benchmark JRip algorithm, PSO/ACO (P) outperforms PSO/ACO (S×P) by 7 (out of a possible maximum value of 16).

Table 3. Accuracy of Labelled Approaches in UCI Data Sets, with Standard Deviation and Student T-Test Shadings

	JRip	**Part**	**PSO/ACO2 (S×P)**	**PSO/ACO2 (P)**
BC	69.95 ±5.66	69.19 ±7.71	71.69 ±6.71	74.16 ±6.47
Crx	85.6 ±4.79	84.4 ±5.34	86.05 ±4.08	86.19 ±4.59
Diabetes	75.39 ±4.87	74.36 ±4.51	71.37 ±8.64	72.67 ±5.28
Heart-c	78.55 ±7.17	76.84 ±7.8	80.14 ±8.02	80.53 ±10.21
Hepatitis	90.42 ±9.38	84.86 ±14.0	80.0 ±15.2	88.19 ±14.14

Table Continued

Table 3. Accuracy of Labelled Approaches in UCI Data Sets, with Standard Deviation and Student T-Test Shadings (continued)

Ionosphere	88.05 ±5.13	90.04 ±4.68	80.64 ±6.77	90.04 ±3.58
Iris-Discretized	93.33 ±5.44	94.0 ±5.84	94.67 ±6.13	97.33 ±4.66
Lymph	78.43 ±8.08	83.19 ±9.47	75.0 ±11.04	79.81 ±6.84
Mushroom	99.98 ±0.08	100.0 ±0.0	98.84 ±0.92	99.74 ±0.51
Promoters	71.91 ±13.33	83.91 ±7.91	77.45 ±13.1	77.45 ±13.92
Sonar	73.4 ±11.35	72.52 ±10.57	53.88 ±10.05	64.0 ±14.56
Splice	94.61 ±1.5	92.79 ±1.65	91.5 ±1.69	94.33 ±1.46
Tic-Tac-Toe	97.5 ±1.0	93.85 ±2.7	73.38 ±3.49	98.64 ±0.7
Vehicle	51.17 ±4.82	58.02 ±6.36	44.69 ±6.95	58.6 ±7.89
Wisconsin	93.99 ±3.22	94.43 ±2.06	93.7 ±2.87	93.85 ±2.67
Zoo	90.18 ±9.23	94.18 ±6.6	95.09 ±7.01	81.36 ±9.1

6. DISCUSSION AND CONCLUSIONS

Our experiments show that the new variant of PSO/ACO2 is at least comparable in terms of accuracy with two leading classification algorithms, JRip (RIPPER) and Part (C4.5Rules). They also show that the version of the PSO/ACO2 algorithm that uses just precision with Laplace correction outperforms the variant that uses sensitivity × precision with Laplace correction. As usual some algorithms perform better in certain situations, but on average PSO/ACO2 with Laplace correction performs the best in these 16 data sets. We believe this is because both measures give a level of trade-off between the generality of the rule within the predicted class and how specific the rule is to the predicted class. In the case of the measure that uses sensitivity it causes the algorithm to more strongly attempt to cover as many examples in the predicted class as possible. It is good to make general rules as rules that are not general are likely to perform badly in the test set. For example, in the most extreme case, a rule for every example in the training set could be created, this would make a very specific rule set, but it it very likely to perform badly in the test set. The opposite is also true with rules that are too general without consideration for the specificity in the predicted class. An extreme example would be a rule with an empty antecedent which would always be very general but would only perform at the default (majority class prediction) accuracy. Using precision alone allows the algorithm to cover small patterns in the data set that are not too small as to be overfitted. Sensitivity × Precision creates rules that are overly general and so causes the less optimal accuracies in the test set.

At present PSO/ACO2 is still greedy in the sense that it builds each rule with the aim of optimising that rule's quality individually, without directly taking into account the interaction with other rules. A less greedy, but possibly more computationally expensive way to approach the problem would be to associate a particle with an entire rule set and then to consider the quality of the entire rule set when evaluating a particle, this is known as the "Pittsburgh approach" in the evolutionary algorithm literature. Also the nominal part of the rule is always discovered first and separately from the continuous part, it could be advantageous to use a more "co-evolved" approach. More extensive comparisons between meta-heuristic algorithms, quality measures and rule induction algorithms are left for future research. Along with different topologies and levels of connectivity within the population.

7. REFERENCES

[1] D. Bratton and J. Kennedy. *Defining a Standard for Particle Swarm Optimization*. Proceedings of the 2007 IEEE Swarm Intelligence Symposium, Honolulu, Hawaii, USA, April 2007.

[2] P. Clark and R. Boswell. *Rule induction with CN2: Some recent improvements*. *Machine Learning - EWSL-91*, pp. 151-163, Berlin, 1991.

[3] U.M. Fayyad, G. Piatetsky-Shapiro and P. Smyth. *From data mining to knowledge discovery: an overview*, Advances in Knowledge Discovery and Data Mining, AAAI/MIT, pp. 1-34, 1996.

[4] D.J. Hand. *Construction and Assessment of Classification Rules*. Wiley, 1997.

[5] N. Holden and A.A. Freitas. *A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data*. In: Proc. 2005 IEEE Swarm Intelligence Symposium (SIS-05), pp. 100-107, IEEE, 2005.

[6] N. Holden and A.A. Freitas. *Hierarchical Classification of G-Protein-Coupled Receptors with a PSO/ACO Algorithm*. In: Proc. IEEE Swarm Intelligence Symposium (SIS-06), pp. 77-84. IEEE, 2006.

[7] J. Kennedy and R. Mendes, *Population structure and particle swarm performance*. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii USA. 2002

[8] J. Kennedy and R. C. Eberhart, with Y. Shi. *Swarm Intelligence,* San Francisco: Morgan Kaufmann/ Academic Press, 2001.1

[9] D.J. Newman, S. Hettich, C.L. Blake and C.J. Merz (1998). *UCI Repository of machine learning databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of California, Department of Information and Computer Science.

[10] R.S. Parpinelli, H.S. Lopes and A.A. Freitas. *Data Mining with an Ant Colony Optimization Algorithm*, IEEE Trans. on Evolutionary Computation, special issue on Ant Colony algorithms, 6(4), pp. 321-332, Aug 2002.

[11] T. Sousa, A. Silva, A. Neves, *Particle Swarm based Data Mining Algorithms for classification tasks*, Parallel Computing 30, pp. 767–783, 2004.

[12] I.H. Witten, E. Frank, *Data Mining: Practical machine learning tools and techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, CA, 2005.