SWARM INTELLIGENCE FOR RULE DISCOVERY IN DATA MINING

Andre B. de Carvalho, Taylor Savegnago and Aurora Pozo
Federal University of Parana, Curitiba PR, Brazil

Keywords: Particle swarm optimization, Rule induction, Multi-objective optimization.

Abstract: This paper aims to discuss Swarm Intelligence approaches for Rule Discovery in Data Mining. The first approach is a new rule learning algorithm based on Particle Swarm Optimization (PSO) and that uses a Multiobjective technique to conceive a complete novel approach to induce classifiers, called MOPSO-N. In this approach the properties of the rules can be expressed in different objectives and then the algorithm finds these rules in an unique run by exploring Pareto dominance concepts. The second approach, called PSO/ACO2 algorithm, uses a hybrid technique combining Particle Swarm Optimization and Ant Colony Optimization. Both approaches directly deal with continuous and nominal attribute values, a feature that current bioinspired rule induction algorithms lack. In this work, an experiment is performed to evaluated both approaches by comparing the performance of the induced classifiers.

1 INTRODUCTION

Data mining is the overall process of extracting knowledge from data. In this area, rules are one of the most used forms to represent the extracted knowledge. This is because of their simplicity, intuitive aspect, modularity, and because they can be obtained directly from a data set (Fawcett, 2001). Therefore, rules induction has been established as a fundamental component of many data mining systems.

This work deals with learning classification rules, or rule induction. In the rule induction process, the algorithm receives a labeled data set as input. This data set must contain examples with the descriptor attributes and the class values. This attributes could have nominal or discrete values. The algorithm looks to the examples and identifies patterns. These patterns are going to be represented in rules. A rule is a pair <antecedent, consequent> or if *antecedent* then *consequent*. The antecedent represents the restrictions in the values of the attributes and the consequent is the class value. After the end of the learning process, new examples that arrive can be classified by the learned rules.

Although many techniques have been proposed and successfully implemented, few works take into account the importance of the comprehensibility aspect of the generated models and rules. Other important challenge in this area is related to the kind of attributes, previous works with Swarm Intelligence, have never addressed the case where the data sets contain both continuous and nominal attributes. Considering this fact, this work discuss two, recently presented, different algorithms based on Swarm Intelligence for Discovering Rules in data mining context, called MOPSO-N (Carvalho and Pozo, 2008) and PSO/ACO2 (Holden and Freitas, 2008). Both approaches directly deal with both continuous and nominal attribute values.

The MOPSO-N (Multi-Objective Particle Swarm Optimization-N) algorithm is our proposed approach and was first presented in (Carvalho and Pozo, 2008). It is based on Particle Swarm Optimization (Kennedy and Eberhart, 1995) (PSO) and uses a multiobjective approach where the properties of the rules can be expressed in different objectives and the algorithm finds these rules in a unique run. These rules can be used as an unordered classifier. The proposed algorithm has the goal to generate a good classifier in terms of the Area Under the ROC Curve, AUC. So, two objectives were chosen, the sensitivity and specificity criteria that are directly related with the ROC curve (Fawcett, 2001). The hypothesis behind these strategies is that classifiers composed by rules that maximize these objectives present good performance in terms of AUC. The MOPSO-N algorithm was validated using the AUC and comparing its performance to other well known traditional rule induction algorithms like C4.5, NNge and RIPPER. In this experiment the algorithm obtained good results, very competitive with the other algorithms.

The PSO/ACO2, was proposed by Holden and Freitas in (Holden and Freitas, 2008), uses a hybrid technique combining Particle Swarm Optimization and Ant Colony Optimization (Dorigo and Stützle, 2004). The algorithm uses a traditional sequential covering approach to discover one rule at a time. This rule is found by a two steps one run of the Ant Colony algorithm searches a rule based on nominal attributes and after then a conventional PSO algorithm with constriction use this rule as a base for the discovery of a rule with continuous values. The algorithm was evaluated on 27 public domain benchmark data sets used in the classification literature, and the authors showed that PSO/ACO2 is at least competitive with PART in terms of accuracy and that PSO/ACO2 often generates a simpler rule sets.

After, the study of these systems, some important differences between both systems can be noticed: the multiobjective versus the sequential covering approaches, and the hybrid PSO/ACO versus the pure PSO to deal with nominal and continuous attributes.

In this work, the main aspects of both algorithms are presented and them an empirical set of experiments is made. This experiments has the goal to compare both Swarm Intelligence algorithms, MOPSO-N and PSO/ACO2. The algorithms are evaluated comparing the performance of the induced classifiers in respect to the accuracy and ROC graph analysis.

The rest of this paper is organized as follows. Section 2 reviews mains concepts of Multiple Objective Particle Swarm and describes the MOPSO-N algorithm. Section 3 gives a brief review of the PSO/ACO2 algorithm. In Section 4 the experiments are discussed. Finally, Section 5 concludes the paper and discusses future works.

2 RULE LEARNING WITH MULTIOBJECTIVE PARTICLE SWARM OPTIMIZATION

This section presents the main aspects of Multiobjective Particle Swarm Optimization and the proposed algorithm, MOPSO-N. First, Section 2.1 discuss some interest topics of the MOPSO metaheuristic. In Section 2.2, the MOPSO-N algorithm is presented.

2.1 Multiobjective Particle Swarm Optimization

Particle Swarm Optimization (Kennedy and Eberhart, 1995) (PSO) works with a population-based heuristic inspired by the social behavior of bird flocking aiming to find food. . In Particle Swarm Optimization the system initializes with a set of solutions and searches for optima by updating generations. The set of possible solutions is a set of particles, called swarm, which moves in the search space, in a cooperative search procedure. These moves are performed by an operator called velocity of a particle and moves it through an n-dimensional space based on the best positions of their leader (social component) and on their own best position (local component).

Optimization problems with two or more objective functions are called Multiobjective. In such problems, the objectives to be optimized are usually in conflict, which means that there is not a single solution for these problems and the goal is to find a good trade-off of solutions that represent the better possible compromise among the objectives. The general multiobjective maximization problem (with no restrictions) can be stated as to maximize (1).

$$\overrightarrow{f}(\overrightarrow{x}) = (f_1(\overrightarrow{x}), ..., f_Q(\overrightarrow{x})) \tag{1}$$

subjected to $\overrightarrow{x} \in \Pi$, where: $\overrightarrow{x}$ is a vector of decision variables and Π is a finite set of feasible solutions. Let $\overrightarrow{x} \in \Pi$ and $\overrightarrow{y} \in \Pi$ be two solutions. For a maximization problem, the solution $\overrightarrow{x}$ dominates $\overrightarrow{y}$ if:

$$\forall f_i \in \overrightarrow{f}, i = 1...Q, f_i(\overrightarrow{x}) \geq f_i(\overrightarrow{y}), and$$
$$\exists f_i \in \overrightarrow{f}, f_i(\overrightarrow{x}) > f_i(\overrightarrow{y})$$

$\overrightarrow{x}$ is a non-dominated solution if there is no solution $\overrightarrow{y}$ that dominates $\overrightarrow{x}$.

The goal is to discover solutions that are not dominated by any other in the objective space. A set of non-dominated solutions is called Pareto optimal and the set of all non-dominated objective vectors is called Pareto Front.

In Multiobjective Particle Swarm Optimization there are many fitness functions. By exploring Pareto dominance concepts, it is possible to obtain results with specific properties. Based on this concept each particle of the swarm could have different leaders, but only one may be selected to update the velocity. This set of leaders is stored in a repository, which contains the best non-dominated solutions found. The MOPSO components are defined as follows.

Each particle p_i, at a time step t, has a position $x(t) \in R^n$, that represents a possible solution. The position of the particle, at time $t+1$, is obtained by adding its velocity, $v(t) \in R^n$, to $x(t)$:

$$\overrightarrow{x}(t+1) = \overrightarrow{x}(t) + \overrightarrow{v}(t+1) \tag{2}$$

The velocity of a particle p_i is based on the best position already fetched by the particle, $\overrightarrow{p}_{best}(t)$, and the best position already fetched by the set of neighbors of p_i, $\overrightarrow{R_h}(t)$, that is a leader from the repository. The velocity update function, in time step $t+1$ is defined as follows:

$$\begin{aligned}\overrightarrow{v}(t+1) = \varpi * \overrightarrow{v}(t) + \\ + (c_1 * \phi_1) * (\overrightarrow{p}_{best}(t) - \overrightarrow{x}(t)) \quad + \\ + (c_2 * \phi_2) * (\overrightarrow{R_h}(t) - \overrightarrow{x}(t)) \quad (3)\end{aligned}$$

The variables ϕ_1 and ϕ_2, in Equation 3, are coefficients that determine the influence of the particle's positions. The constants c_1 and c_2 indicates how much each component influences on the velocity. The coefficient ϖ is the particle inertia and controls how much the previous velocity affects the current one. $\overrightarrow{R}_h$ is a particle from the repository, chosen as a guide of p_i. There are many ways to make this choice. At the end of the algorithm, the solutions in the repository are the final output. One possible way to make the leader choice is called the sigma distance (Mostaghim and Teich, 2003).

2.2 MOPSO-N Algorithm

MOPSO-N was proposed to handle with both numerical and discrete attributes. In this way, it can be used in different domains, mainly those ones with continuous attributes. It was first introduced in (Carvalho and Pozo, 2008). In this section, MOPSO-N aspects, such as representation and generation of the particles are described.

The algorithm uses the Michigan approach where each particle represents a single solution or a rule. In this context, a particle is an *n*-dimensional vector of real numbers. One integer number represents the value for each discrete attribute and two real numbers represent an interval for numerical attributes. The interval is defined by its lower and upper values. Each attribute can accept the value '?', which means that, for that rule, the attribute does not matter for the classification. In the proposed approach the class value is set in the beginning of the execution. To represent the particle as a possible solution, the attributes values must be codified into real numbers. The codification of the discrete attributes is conceived by real numbers related to each attribute value of the database. When a numerical attribute has a void value, their cells in the particle representation receive the lower bound value available on the database.

The rule learning algorithm using MOPSO-N works as follows. The initialization procedure randomly spreads all particles in the search space and initializes the components of the particle. In this process, the discrete attributes are defined by using a roulette procedure, where the most frequent values have higher probabilities. The probability of the generic value of each attribute, '?', is a function of the number of possible values for the attribute. For the numerical attributes, first, all attributes have the probability to be empty, `prob_empty`. If an attribute is set as non-empty, the lower and upper limits are spread randomly in the interval defined by the `minimum` and `maximum` values for the attribute (obtained from the data set). After then, the particle's components, like velocity and local leader are initialized.

Once the initial configuration is performed, the evolutionary loop is executed performing the moves of all particles in the search space. In this work the stop criterion is the maximum number of generations. In each iteration, initially, the operations discussed in the previous section are implemented.

In the position update, a `mod` operator is applied. This operation is used to limit the particle into the search space and to to promote equal probability of selection to each attribute values. For discrete attributes, the values are bounded to the number of values for each attribute. For the numerical ones, a `mod` operator is proposed. It is executed using the maximum and minimum values of the attribute. If the new upper limit overflows the maximum value, the excess is added to the minimum value, and that is the new limit. After this process, the smaller value is the new lower limit, and the larger is the upper. If both values overflow the limits, the attributes are set to empty ('?').

After all particles have been moved through the search space, they are evaluated using the objectives and once again the best particles are loaded in the repository and the global leaders are redefined. A procedure checks for more general or specific rules in the repository before a rule is added to it. A rule is more specific than other rule if it has less attribute constraints and the same contingency table. Only the more general rules are kept in the archive.

At the end of the execution, the rules are usually aggregated into a rule set to build a classifier. After, the classifier can be used to classify unseen instances. The classification of a new instance is performed by a voting process. In the voting process, all rules vote the class of the instances. The process contains the following steps: for each class, all rules that cover the input instance are identified. The identified rules are sorted according to some ordering criteria. This work uses the Laplace Accuracy, discussed in (Yanbo J. Wang and Coenen, 2006). The ordering process is applied to allow the selection of only the best *k* rules, according to the ordering criteria, to vote the class of the instance. This number can vary from one rule to all rules. The next step counts the votes of each class. Finally, the class that scores higher is declared as the class of the example.

3 THE PSO/ACO2 ALGORITHM

The PSO/ACO2 (Holden and Freitas, 2008) algorithm achieves a native support of nominal data by combining ideas from ant colony optimization (Dorigo and Stützle, 2004) and particle swarm optimization (Kennedy and Eberhart, 1995) to create a classification metaheuristic that supports innately both nominal and continuous attributes.

The algorithm uses a traditional sequential covering approach to discover one classification rule at a time. The sequential covering approach starts with an empty set of rules. Then, for each class the algorithm performs a WHILE loop, where TS is used to store the set of training examples the rules will be created from. On each iteration of this loop, one run of the ant colony algorithm finds a rule based on nominal attributes. This rule is used as a base for the discovery of a rule with continuous values. For the continuous part of the rule, a conventional PSO algorithm with constriction is used. The PSO algorithm returns the best discovered rule (BestRule). After the BestRule has been generated it is then added to the rule set, but, after a pruning procedure inspired by Ant-Miner's pruning procedure (Parpinelli et al., 2002). Finally, the algorithm removes all the examples covered by the rule from TS and the process is repeated with the remaining examples, and continues until the maximum number of uncovered examples per class (MaxUncovExampPerClass) is reached. When this threshold has been reached TS is reset by adding all the previously covered examples. This process means that the rule set generated is unordered and is submitted to an ordered process, considering a quality measure. The measure used to estimate the quality of a rule is the Laplace-corrected Precision. Also, after the rule set has been ordered it is pruned as a whole. The pruned process involves tentatively to remove terms from each rule and seeing if each term's removal affects the accuracy of the entire rule set. After this process is complete, the algorithm also removes whole rules that do not contribute to the classification accuracy.

The algorithm was evaluated on 27 public domain benchmark data sets used in the classification literature, and the authors shown that PSO/ACO2 is at least competitive with PART, an industry standard classification algorithm, in terms of accuracy, and that PSO/ACO2 often generates much simpler rule sets. However, the authors also comment that PSO/ACO2 is partly greedy in the sense that it builds each rule with the aim of optimizing that rule's quality individually, without directly taking into account the interaction with other rules.

Table 1: Description of data sets used in the experiments.

#	Data set	Attributes	Examples	% of Majority Class
1	breast	10	683	65.00
2	bupa	7	345	57.97
3	ecoli	8	336	89.58
4	flag	29	174	91.23
5	glass	10	214	92.05
6	haberman	4	306	73.52
7	heart	14	270	55.00
8	ionosphere	34	351	64.10
9	new-thyroid	6	215	83.72
10	pima	9	768	65.51
11	vehicle	19	846	76.47

4 EXPERIMENTS

This Section describes an experiment to compare the two Swarm Intelligence algorithms, MOPSO-N and PSO/ACO2. The classification results of MOPSO-N, PSO/ACO2 are evaluated using the accuracy measure and through ROC Graph analysis. Section 4.1 presents the methodology used in the experiment and Section 4.2 presents the results.

4.1 Methodology

For the experiments, both algorithms were evaluated through 11 databases from the UCI machine learning repository (Asuncion and Newman, 2007)(Table 1). The experiments were executed using 10-fold-stratified cross validation and for all algorithms were given the same training and test files.

In the experiments two comparison were made: comparison of the accuracy and a ROC Graph analysis. The accuracy (4) relates the total number of instances correctly classified, True Positive (TP) and True Negative (TN), to the total number of instances(N_{inst}).

$$Accuracy = \frac{TP + TN}{N_{inst}} \qquad (4)$$

The non-parametric Wilcoxon test, with 5% confidence level, was executed to define which algorithm is better. It was adopted the methodology presented in (Demšar, 2006). Here, the Wilcoxon test was applied with all accuracy values for all data sets.

Table 2: Average values of accuracy and number of rules for each algorithm.

Data set	MOPSO-N		PSO/ACO2	
	Accuracy	# Rules	Accuracy	# Rules
1	95.18 (1.38)	80.51	96.62 (1.95)	7.4
2	65.78 (5.63)	164.97	69.29 (6.30)	22.3
3	90.60 (32.09)	60.63	89.26 (5.91)	5.6
4	90.76 (3.38)	53.59	89.21 (4.45)	5.4
5	91.61 (16.22)	57.73	90.23 (5.50)	5.4
6	75.30 (10.50)	185.97	73.19 (9.03)	13.2
7	80.00 (8.93)	162.06	80.00 (10.50)	11.4
8	85.14 (6.55)	66.6	83.75 (7.01)	3.9
9	93.52 (9.89)	36.8	94.93 (3.97)	2
10	74.18 (4.59)	243.46	73.95 (6.21)	32.4
11	85.58 (3.2)	142.02	95.02 (2.43)	14.4

The second comparison has the goal to compare the classification of the Swarm Intelligence algorithms in terms of a ROC Graph analysis. This analysis is considered a relevant criterion to deal with imbalanced data, misclassification costs and noisy data. The ROC Graph relates the false positive rate, fpr, (axis-x) and the true positives rate, tpr, (axis-y) of a classifier (Fawcett, 2001).

To compare the classification results of classifiers, often it is used the Area Under the ROC Curve (AUC). However the PSO/ACO2 implementation used do not return this measure. To overcome this limitation, in this experiment was performed a multiobjective analysis based on the ROC Graph. To determine which algorithm has better performance it was applied the dominance ranking (Zitzler and Thiele, 1999). Dominance ranking stands for a general, preference independent assessment method that is based on pairwise comparisons. Here, each execution of both algorithm was compared. In this analysis, for each fold of data set, the execution of both algorithms is compared, and the test indicates which algorithm dominates the other, i.e., has greater values of tpr and lower values of fpr. After the execution of the dominance ranking between MOPSO-N and PSO/ACO2, the Wilcoxon test was performed to define which algorithm has the best ranking.

The MOPSO-N parameter are: 100 generations, 500 particles, for each fold, one run was performed, ω, ϕ_1 and ϕ_2, randomly vary $[0,0.8]$, $[0,1] and [0,1]$, respectively, c_1 and c_2 were defined as 2.05 and texttprob_empty was empirically set to 0.1.

The PSO/ACO2 classification algorithm is freely available[1] and was executed with the default parameters presented in (Holden and Freitas, 2008). There, the authors used 100 iterations and 100 particles, for each call to the PSO algorithm, and the fitness function adopted was the precision of the rule.

[1]Available at http://sourceforge.net/projects/psoaco2.

Table 3: Average values of tpr and fpr for each algorithm.

Data set	MOPSO-N		PSO/ACO2	
	tpr	fpr	tpr	fpr
1	87.90 (9.89)	2.46 (2.66)	97.53 (2.66)	5.00 (4.73)
2	32.28 (11.47)	10.00 (8.81)	61.95 (16.21)	25.50 (9.55)
3	14.16 (19.26)	0.00 (0.00)	95.84 (3.76)	57.50 (23.38)
4	5.00 (15.81)	0.58 (1.86)	97.71 (2.95)	100.00 (0.00)
5	5.00 (15.81)	1.55 (3.52)	97.71 (2.95)	100.00 (0.00)
6	24.72 (13.20)	4.92 (5.37)	86.71 (7.52)	64.86 (23.76)
7	75.83 (15.44)	17.33 (10.97)	72.50 (11.81)	13.33 (13.69)
8	67.37 (15.06)	2.64 (4.26)	92.43 (6.61)	31.79 (19.30)
9	64.16 (30.18)	0.00 (0.00)	98.88 (3.51)	8.33 (18.00)
10	93.20 (3.55)	60.11 (12.10)	61.86 (11.20)	19.60 (5.94)
11	40.81 (13.14)	0.62 (0.80)	87.97 (6.71)	2.79 (2.05)

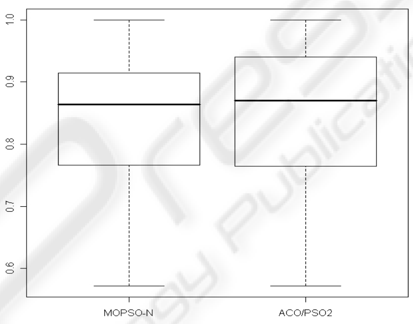

Figure 1: Box plot of the accuracy values for MOPSO-N and PSO/ACO2 for all data sets.

4.2 Swarm Intelligence Algorithms Comparison

The average values of accuracy and number of rules (average of 10 folds values) for each data set are presented in Table 2. The number between brackets indicates the standard deviation.

For the Wilcoxon test, the p-value obtained was 0.3938, so both algorithms have equivalent results. Figure 1 shows the box plot of the accuracy values used in the test. This result shows that the MOPSO-N algorithm has good classification results in terms of accuracy, although it has the main concern to generate good values of AUC. However the PSO/ACO2 generates a smaller number of rules than MOPSO-N. This characteristic is attractive, since a simple classifier can be more understandable.

For the second comparison the values for tpr and fpr for all data sets were evaluated through the dominance rank test. Table 3 presents the average value of tpr and fpr (average of 10 folds values) for all data sets obtained for each algorithm.

In this experiment the MOPSO-N algorithm obtained better results than PSO/ACO2, according to Wilcoxon test, with p-value = 0.0013. It can be observed that MOPSO obtained very good values of

fpr, even when dealing with unbalanced data. As example, for data set 4, MOPSO obtained $fpr = 0.58$, while PSO/ACO2 obtained $fpr = 100$, i.e., while MOPSO makes few mistakes, the PSO/ACO2 classifies almost every instance as the positive class. The PSO/ACO2 algorithm obtained better values of tpr than MOPSO-N, i.e., PSO/ACO2 makes more hits then MOPSO-N, however these hits are obtained through a classification that produces a high number of errors. These very good values of fpr made the MOPSO algorithm had good results in the dominance ranking test, since their results are rarely dominated by the PSO/ACO2 results.

These results stress that our approach has good classification results when analyzing the results through the ROC Graph. This confirms the hypothesis that maximizing the Sensitivity and the Specificity of rules will generate a good classifier.

5 CONCLUSIONS

This work explores the Swarm Intelligence in rule learning context. Two different algorithms based on these concepts are discussed: MOPSO-N and PSO/ACO2. The MOPSO-N algorithm has the following properties: first, MOPSO-N uses a Multiobjective approach that allows us to create classifiers in only one execution. This method works finding the best non-dominated rules of a problem, by selecting them in the rules generation process. Second, the algorithm deals with both numerical and discrete data. The PSO/ACO2 explores a hybrid approach using a sequential covering algorithm that combines the ant colony optimization technique to learn the nominal part of a rule, and the particle swarm optimization technique to deal with the continuous part of a rule.

Then, an empirical study was made to compare these two approaches. Both algorithms are equivalent in terms of accuracy, but the MOPSO-N outperforms the PSO/ACO2 in ROC Graph Analysis. The PSO/ACO2 algorithm has the advantage to generate a smaller set of rules and them to generate a more comprehensive model. Also, this study complements the results presented in (Carvalho and Pozo, 2008) and showed that the MOPSO-N algorithm has also good classifications results when compared to a bioinspired algorithm.

Future works include: the execution of a greater number of experiments to validate the initial results, the comparison of our approach with other known algorithms in the literature, and enhancements of the algorithm to profit a more diverse set of rules without increasing the size of the rule set, for example, the prune procedure implemented by the PSO/ACO2 algorithm must be investigated.

REFERENCES

Asuncion, A. and Newman, D. J. (2007). UCI Machine Learning Repository, [http://www.ics.uci.edu/∼mlearn/MLRepository.html]. Irvine, CA: University of California, School of Information e Computer Science.

Carvalho, A. B. and Pozo, A. (2008). Non-ordered data mining rules through multi-objective particle swarm optimization: Dealing with numeric and discrete attributes. In *Poceedings of Hybrid Intelligent Systems, 2008. HIS '08. Eighth International Conference*, pages 495–500.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.*, 7:1–30.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimization*. The MIT Press.

Fawcett, T. (2001). Using rule sets to maximize ROC performance. In *IEEE International Conference on Data Mining*, pages 131–138. IEEE Computer Society.

Holden, N. and Freitas, A. A. (2008). A hybrid pso/aco algorithm for discovering classification rules in data mining. *Journal of Artificial Evolution Applications*, 2008(3):1–11.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE Press.

Mostaghim, S. and Teich, J. (2003). Strategies for finding good local guides in multi-objective particle swarm optimization. In *SIS '03 Swarm Intelligence Symposium*, pages 26–33. Proceedings of the 2003 IEEE Swarm Intelligence Symposium. IEEE Computer Society.

Parpinelli, R., Lopes, H., and Freitas, A. (2002). Data Mining with an Ant Colony Optimization Algorithm. *IEEE Trans on Evolutionary Computation, special issue on Ant Colony Algorithms*, 6(4):321–332.

Yanbo J. Wang, Q. X. and Coenen, F. (2006). A novel rule ordering approach in classification association rule mining. *International Journal of Computational Intelligence Research*, 2(3):287–308.

Zitzler, E. and Thiele, L. (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271.