Classification Rule Mining with Iterated Greedy

Juan A. Pedraza[1], Carlos García-Martínez[2], Alberto Cano[2], and Sebastián Ventura[2]

[1] I+D Dpt., Yerbabuena Software,
Málaga 29590, España, `jantpedraza@gmail.com`
[2] Dpt. of Computing and Numerical Analysis, University of Córdoba,
Córdoba 14071, España, `cgarcia@uco.es`,`acano@uco.es`,`sventura@uco.es`

Abstract. In the context of data mining, *classification rule discovering* is the task of designing accurate rule based systems that model the useful knowledge that differentiate some data classes from others, and is present in large data sets.
Iterated greedy search is a powerful metaheuristic, successfully applied to different optimisation problems, which to our knowledge, has not previously been used for classification rule mining.
In this work, we analyse the convenience of using iterated greedy algorithms for the design of rule classification systems. We present and study different alternatives and compare the results with state-of-the-art methodologies from the literature. The results show that iterated greedy search may generate accurate rule classification systems with acceptable interpretability levels.

Keywords: Classification rule mining, iterated greedy, interpretability

1 Introduction

Data mining [1] involves the use of data analysis tools to discover useful knowledge from large data sets. Classification is a form of data analysis that extracts models describing important data classes by means of their properties. Classification has been successfully applied to several fields such as medical diagnosis or credit risk evaluation, among others, by means of several approaches of very different nature, for instance, artificial neural networks, support vector machines, or instance-based techniques [2].

Rule-based systems [3] are a representation paradigm that models the knowledge of a specific area of interest by means of sets of rules with an antecedent and one consequent. When applied to classification problems, the antecedents of the rules define some property relations, which are oftenly present in particular sets of patterns, and the consequents emit predictions of the class they belong. Comprehensibility is one of the benefits that rule-based systems possess and have lately attracted the attention of the research community [4–7].

Iterated greedy search (IG) is a simple yet effective metaheuristic that has been successfully applied to different combinatorial problems [8–12]. IG explores the solution search space by iterating over a greedy process applied to a single solution and composed by two main phases: *destruction* and *construction*. During the destruction phase, some solutions components are removed, producing a partial solution. Afterwards, the construction phase applies a greedy heuristic to complete this partial solution.

To our knowledge, IG has not previously been applied to the design of rule-based classification systems. In this work, we are interested in analysing the possibility and benefits of generating rule classification systems according to the iterative IG framework. Consequently, we address it as a combinatorial problem where a solution represents a rule based system and undergoes through an iterative process of destruction and construction.

The rest of this work is structured as follows. In Section 2, we address the adaptation of the IG metaheuristic to the problem of generating good rule classification systems. In Section 3, we present several empirical studies aimed at: 1) analysing the influence of the parameters and settings associated with the method that provide accurate rule classification systems, and 2) comparing the resulting IG algorithm for classification rule mining (IG-RMiner) with other prominent approaches from the literature in terms of accuracy and interpretability. Finally, in Section 4, we discuss conclusions and future work.

2 An IG model for rule mining

In this section, we describe our adaptation of the IG model for rule mining. The main framework is presented in Section 2.1, and the alternatives for the construction phase, in Section 2.2.

2.1 General scheme of the IG for rule mining

Figure 1 depicts the outline of our IG metaheuristic for classification rule mining (IG-RMiner). It starts from a single initial solution, a rule-based classification system, generated from scratch by an heuristic construction procedure (steps 1-3; Section 2.2). Afterwards, it iterates through a main loop in which first, a partial solution S_d is obtained at the destruction phase (step 5), and second, a complete candidate solution S_c is reconstructed by applying the same construction procedure to S_d (step 6). A pruning phase is added to improve the constructed solution, if possible (step 7). Before continuing with the next iteration, an acceptance criterion decides whether the solution returned by the pruning phase, S_p, becomes the new current solution (step 11). The process iterates until some termination conditions have been met (e.g. maximum number of iterations, or maximum computation time allotted). The best solution, S_b, generated during the iterative process is kept as the overall result.

The specific features of the IG-RMiner are:

- The method starts with an empty solution with one rule per class and no conditions in their antecedents. When classifying a new pattern, conflicts are resolved according to the rule with higher accuracy.

```
Input:
 Greedy-construction(·): Greedy construction procedure
 Acceptance-criterion(·,·): Acceptance criterion
 Stop-condition: Stop condition
 $p_d$: Probability for removing conditions
Output:  $S_b$: Best solution generated
$S_d \leftarrow \emptyset$;
$S \leftarrow$ Greedy-construction($S_d$);
$S_b \leftarrow S$;
while Stop-condition is not reached do
    $S_d \leftarrow$destruction($S, p_d$);
    $S_c \leftarrow$ Greedy-construction($S_d$);
    $S_p \leftarrow$ pruning($S_c$);
    if $S_p$ is better than $S_b$ then
        $S_b \leftarrow S_p$;
    end
    $S \leftarrow$ Acceptance-criterion($S, S_p$);
end
return $S_b$;
```

Fig. 1. Pseudocode of the IG-RMiner model

– The heuristic construction procedure intents to improve the quality of the rules by appending new conditions, one by one, to their antecedents. Section 2.2 specifies the analysed alternatives for selecting the rule to be improved, the condition to be added, and the quality measure to be considered. This phase iterates until there is not any other condition able to improve the quality measure.

– The standard destruction mechanism of IG removes a percentage of random components of the current solution. In our case, the destruction removes a percentage of conditions from some rules, at random.

– The pruning phase revises the conditions in the antecedents of the rules checking whether their individual extraction might improve the global accuracy of the system. If this is the case, the revised condition is removed. This situation may occur, for instance, when the construction phase has included several conditions in the rules step by step, but the last included ones interfere with some others previously inserted.

– Two acceptance criteria, commonly applied in standard IGs, have been studied in this work: replace if and only if the new solution is better than the current one (RB - replace if better) [13], and replace always (RA) [9, 14, 15]. This latter produces higher diversification in the search process carried out by the algorithm.

– The rule based system that is returned at the end is the one with the highest percentage of correctly classified patterns.

2.2 Construction phase

Given a partial solution, i.e., a rule system, with some conditions in the antecedents of its rules, this phase intends to improve the quality of the system and/or its rules by iteratively appending new conditions, until no condition producing an improvement is found. To carry out this goal, we have identified three task to be addressed, and analysed different strategies for each one:

Selection of the rule to be improved: We have studied the performance gained when the algorithm tries to optimise either the rule with the lowest accuracy (LAR - lowest accuracy rule) or a randomly selected one (RR - random rule).

Selection of the condition to be inserted: Once the rule has been selected, the set of compatible conditions (those that do not contradict the current antecedent of the rule) is examined. The grammar in Figure 2 specifies the possible conditions to be generated. Then, two strategies are analysed:

– *Best improvement* (BI): This strategy appends the condition that yields the best improvement in the rule, if this condition exists.
– *First improvement* (FI): This strategy scans through the compatible conditions, in a random order, and chooses the first one that improves the quality of the rule, if this condition exists.

$$\langle S \rangle \rightarrow \langle condition \rangle$$
$$\langle condition \rangle \rightarrow \langle op\_num \rangle \langle var\_num \rangle \langle value \rangle$$
$$\langle condition \rangle \rightarrow \langle op\_nom \rangle \langle var\_cat \rangle \langle value \rangle$$
$$\langle op\_num \rangle \rightarrow \leq \mid \geq$$
$$\langle op\_cat \rangle \rightarrow = \mid \neq$$
$$\langle var\_num \rangle \rightarrow \text{Any valid numerical attribute in data set}$$
$$\langle var\_cat \rangle \rightarrow \text{Any valid categorical attribute in data set}$$
$$\langle value \rangle \rightarrow \text{Any valid corresponding value}$$

Fig. 2. Grammar used to create single attribute-value conditions

Evaluation of the quality improvement: We have studied three quality criteria to evaluate whether the insertion of one condition in the antecedent of the selected rule is or is not favourable (only one at a time):

– *Global accuracy Improvement with penalisation for errors* (GI): This measure is defined as the percentage of correctly classified patterns minus the percentage of incorrectly classified ones. Initial experiments reported poor results if only the percentage of correctly classified patterns was evaluated, so this correction was analysed in the study. According to this criterion, new accepted conditions necessarily result in an accuracy improvement of the rule classification system, and no condition that reduced the global accuracy, with regard to the current state, would be inserted.
– *Rule accuracy Improvement* (RI): Rule accuracy is defined as the percentage of patterns correctly classified by the rule, i.e., those covered by its antecedent that belong to the predicted class (true positives - TP) and those not covered that belong to a different class (true negatives - TN) (Equation 1; NP is the number of patterns). This metric is the common accuracy measure for binary classification [16, 17]. According to this, new conditions result in an accuracy improvement of the rule, but not necessarily of the global system. Even though, the iterative alternation between construction and destruction phases of IG, with the intention of getting accurate rules, might produce more accurate systems. That is the reason why this criterion is studied.
– *Sensitivity-Specificity Multiplication* (SS): Given a binary classifier (for instance, one rule of our system), its sensitivity is defined as the percentage of correctly classified positive patterns (Equation 2); and its specificity is the percentage of correctly classified negative patterns (Equation 3). Both measures should be maximised. Then, the quality of the condition to be inserted is evaluated as the improvement on the multiplication of both metrics, as it is done in other works [4, 18].

$$\text{Rule accuracy} = \frac{TP + TN}{NP} \tag{1}$$

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{2}$$

$$\text{Specifity} = \frac{TN}{TN + FP} \tag{3}$$

Therefore, up to 12 combinations of decision strategies for the construction phase, shown in Table 1, are analysed in this work. For instance, the combination referred to by RR-FI-RI accepts the first condition that appended to a randomly chosen rule, improves the accuracy of that rule.

3 Experiments

We have implemented the different IG-RMiner versions in Weka [19] and performed experiments on an Intel Core i7-930 quad-core 2.80GHz computer with 12GB RAM (only one thread per run). Table 2 shows the characteristics of the used 22 datasets from the UCI [20].

Table 1. Construction strategies combinations

			Quality evaluation		
			GI	RI	SS
Rule	LAR	BI	1. LAR-BI-GI	2. LAR-BI-RI	3. LAR-BI-SS
/		FI	4. LAR-FI-GI	5. LAR-FI-RI	6. LAR-FI-SS
Condition	RR	BI	7. RR-BI-GI	8. RR-BI-RI	9. RR-BI-SS
		FI	10. RR-FI-GI	11. RR-FI-RI	12. RR-FI-SS

Table 2. Used datasets

Name	**# Patterns**	**# Attributes**	**#Classes**
1.Australian	690	14	2
2.Balance-scale	625	4	3
3.Breast-cancer	286	9	2
4.Bupa	345	6	2
5.Car	1728	6	4
6.Chess	3196	36	2
7.Contraceptive	1473	9	3
8.Dermatology	366	34	6
9.Flare 2	1066	11	6
10.German	1000	20	2
11.Haberman	306	3	2
12.Ionosphere	351	33	2
13.Iris	150	4	3
14.Lymph	148	18	4
15.Page-blocks	5472	10	5
16.Segment-challenge	2310	19	7
17.Sonar	208	60	2
18.Tae	151	5	3
19.Thyroid	7200	21	3
20.Tic-tac-toe	958	9	2
21.Vehicle	846	18	4
22.Zoo	101	16	7

Non-parametric tests have been used to compare the results of different algorithms or instances [21]. Specifically, we have considered two alternative methods to analyse the experimental results:

– *Iman and Davenport's test* [22] and *Holm's method* [23] as a post hoc procedure. The first test is used to see whether there are significant statistical differences among the results of a certain group of classifiers. If differences are detected, then, Holm's test is employed to compare the best classifier (control classifier) with the remaining ones.
– *Wilcoxon matched-pairs signed-ranks test* [24], which compares the results of two algorithms directly.

In Section 3.1, we address the setting and election of the parameters and decision strategies of IG-RMiner that result in better classifiers. In Section 3.2, we compare the characteristics of the best IG-RMiner instance with regards to state-of-the-art classification techniques.

3.1 Parameter tuning

Here, we intend to find the combinations, under a full factory design, of acceptance criteria ({RB,RA}), construction components combination (Table 1), and percentage of destructed conditions ($p_d \in \{10\%, 25\%, 50\%\}$) of IG-RMiner, that capacitate it to generate classifiers with high accuracy. The algorithm instances are tested using 5-fold cross-validation on the data sets aforementioned with 5, 50 and 120 seconds as stop conditions for data sets with less than 10, 30, or more attributes, respectively (which are values comparable to those of some algorithms in the following section).

Table 3 shows the mean accuracy levels reached by the best ten IG-RMiner instances (out of 72) on 5 independent runs and all the data sets, together with their mean ranking values. Though Holm's procedure did not find significant differences between the best ranked variant and many others, we observe some commonalities among the best instances:

Table 3. Avg. accuracy, mean ranking and Holm's test on the IG-RMiner variants

IG-RMiner instance	Accuracy	Ranking
IG-RMiner(50%,RR-BI-SS,RA)	**78.2309**	**11.2500**
IG-RMiner(50%,RR-FI-RI,RA)	77.6537	12.1818
IG-RMiner(50%,RR-FI-SS,RA)	77.7715	12.7500
IG-RMiner(50%,RR-BI-RI,RA)	76.9077	15.5455
IG-RMiner(25%,RR-BI-SS,RA)	76.1675	16.8864
IG-RMiner(25%,RR-FI-SS,RA)	76.2823	17.5455
IG-RMiner(25%,RR-FI-RI,RA)	75.9686	18.1136
IG-RMiner(50%,LAR-FI-SS,RA)	72.4367	19.1591
IG-RMiner(50%,LAR-FI-RI,RA)	71.4831	20.4773
IG-RMiner(25%,LAR-FI-SS,RA)	73.0086	20.5000
...	...	...

- All the best variants iterate always from the most recent solution, replacing always the current solution (RA), instead of from the best one.
- Seven variants, the best ranked ones, select a random rule to be improved at the construction phase (RR), not the one with lowest accuracy. On the contrary, there does not seem to exist a clear preference for the condition to be inserted (BI or FI), nor the quality criterion to be evaluated. Regarding this latter, no algorithm between these ten variants considers the global accuracy improvement (GI).
- Most the variants, and the ones in better positions, destroy 50% of the conditions of the rule system per iteration.

Having noticed that the best ranked algorithm, IG-RMiner(50%,RR-BI-SS,RA), satisfies all of previous conclusions, we will compare it with other significant classification techniques in terms of accuracy, interpretability and learning time. From now on, this instance will be referred to by just IG-RMiner.

3.2 Comparison with salient classification techniques

In this Section, we compare IG-RMiner with 10 other significant classification techniques, covering among others, evolutionary based techniques, ant colony optimization, fuzzy rule systems, and classic decision tree generators: ICRM [4], MPLCS [25], ILGA [26], CORE [27], SLAVE [28], GFS-GP [29], DTGA [30], AntMiner+ [31], RIPPER [32], C45R [33]. In particular, ICRM recently proved to be able to generate sufficiently accurate and easily interpretable rule classification systems. In this study, we will analyse the reached accuracy levels of the algorithms, the consumed computation time, and several measures that assess how interpretable the generated systems are, namely, the number of rules, number of conditions (global and mean per rule), and complexity metric [34]. This latter computes the ratio between the number of classes covered and conditions of the rule system (Equation 4):

$$\text{complexity} = \frac{m}{\sum_{i=1}^{r} n_i} \tag{4}$$

where m is the number of classes, r is the number of rules, and n_i is the number of conditions used in the ith rule. This measure returns the value 1 when the classifier contains one rule per class using only one condition each and it approaches 0 if there are more rules and conditions. If there are less rules than classes or there exists a default rule without conditions, this measure can be higher. In any case, we have limited the result to 2 times the number of classes.

For these experiments, IG-RMiner was tested using 10-fold cross-validation, the time limits showed in [4] for ICRM, and 5 independent runs. This setting is in accordance with the experimentation in [4], from where the results of previous algorithms were obtained.

Table 4 and Figure 3 summarise the results. The former shows the algorithms and their mean ranking values ordered, over all the datasets and runs, per performance measure. Statistical performance differences between IG-RMiner and the corresponding algorithm according to the Wilcoxon's test and 5% as significance factor are presented in italics with the character '*' at the beginning. IG-RMiner is highlighted in boldface for reference purposes. When IG-RMiner appears higher in the table, that means that IG-RMiner is statistically better than the corresponding algorithm, and vice versa. More detailed results can be consulted at http://www.uco.es/grupos/kdis/kdiswiki/index.php/IG-RMiner.

From the results in Table 4, we can see that IG-RMiner is able to generate accurate rule classification systems very fast, because it is among the best algorithms with regards to accuracy and time and the Wilcoxon's test does not find significant differences with regards to the best one. Additionally, IG-RMiner obtains competitive results in the interpretability performance measures, being significantly outperformed just by ICRM. The worst results are located in the

Table 4. Algorithms' ranking values per performance measure

Accuracy		**Time**		**#Rules**	
MPLCS	3.08	**IG-RMiner**	**3.04**	**IG-RMiner**	**2.48**
DTGA	4.42	C45R	3.29	ICRM	2.56
IG-RMiner	**4.46**	RIPPER	3.56	_*CORE_	_3.79_
C45R	4.88	DTGA	4.29	_*SLAVE_	_4.63_
ICRM	5.08	_*ICRM_	_4.5_	_*AntMin+_	_4.65_
RIPPER	5.46	_*AntMin+_	_7.1_	_*MPLCS_	_5.06_
AntMin+	6.96	_*SLAVE_	_7.46_	_*C45R_	_7.04_
ILGA	7.67	_*MPLCS_	_7.75_	_*RIPPER_	_8.13_
SLAVE	7.73	_*GFS-GP_	_7.96_	_*ILGA_	_8.5_
GFS-GP	7.9	_*CORE_	_8.33_	_*DTGA_	_8.71_
*CORE	_8.38_	_*ILGA_	_8.71_	_*GFS-GP_	_10.46_

#Conds		**#Conds/#Rules**		**Complexity**	
*ICRM	_1.79_	_*ICRM_	_1.9_	_*ICRM_	_1.43_
IG-RMiner	**3.92**	SLAVE	5.17	**IG-RMiner**	**3.32**
CORE	4.38	CORE	5.29	_*CORE_	_4.26_
SLAVE	4.52	AntMin+	5.33	AntMin+	4.56
AntMin+	4.56	C45R	5.58	_*MPLCS_	_5.02_
*MPLCS	_5.25_	MPLCS	6,08	SLAVE	6.06
_*C45R_	_6.29_	**IG-RMiner**	**6.19**	_*C45R_	_6.23_
*RIPPER	_7.33_	RIPPER	6.29	_*RIPPER_	_7.43_
*DTGA	_8.75_	GFS-GP	7.31	_*DTGA_	_8.58_
*ILGA	_9.04_	_*DTGA_	_7.81_	_*ILGA_	_8.97_
*GFS-GP	_10.17_	_*ILGA_	_9.04_	_*GFS-GP_	_10.08_

ratio between the number of conditions and number of rules. We have identified that this is due to IG-RMiner tends to create complex rules, with many conditions, in a few data sets where there are not many classes. Additionally, we observe that MPLCS and DTGA are the most accurate methods but they obtain poor interpretable performance values, and ICRM provides the most interpretable systems, not much less accurate, but requiring a bit longer times. Therefore, IG-RMiner stands in an intermediate position.

From the visual representation in Figure 3, apart from the clear bias to interpretable classifiers because of the number of associated measures, we observe that ICRM and IG-RMiner are the two algorithms with the smallest associated areas, which is better. The other algorithms have larger associated areas, because either they generally get worse ranking values, or, although obtaining better accuracy values, their results in the interpretability metrics are inferior.

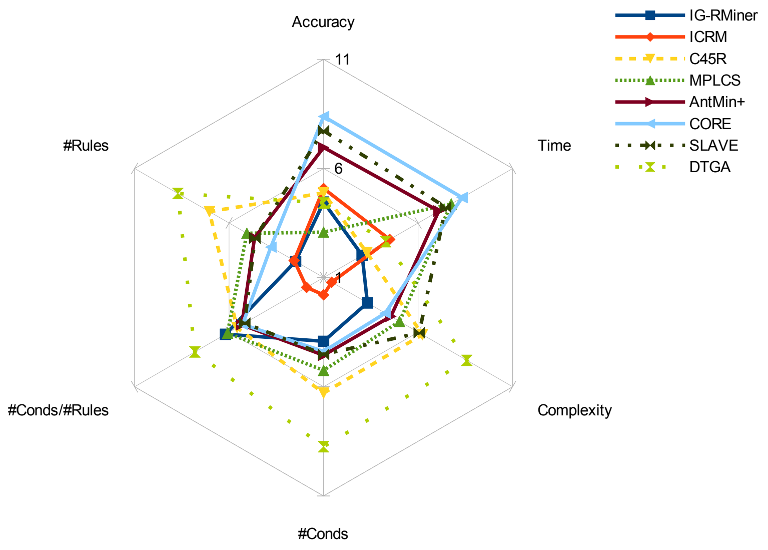

Fig. 3. Algorithms' rankings per performance measure. RIPPER, ILGA, GFS-GP are not represented to make the graph clearer, because they did not get better ranking values than IG-RMiner in neither accuracy nor any interpretability metric

4 Conclusions

We have studied the application of the IG metaheuristic to the problem of designing rule classification systems. Different alternatives have been tested for the construction phase and acceptation criterion of the IG. The result is a technique that constructs accurate rule classification systems with acceptable interpretability levels, with regards to 10 other methodologies from the literature on 22 data sets from the UCI. In particular, the obtained IG-RMiner algorithm iterates through a process in which conditions are randomly destroyed and inserted into the antecedents of random rules to enhance their sensitivity and specificity values.

In our opinion, this line of research is worthy of further studies. We intend to explore the following avenues of research: 1) to include some interpretability criteria in the search process of IG-RMiner in order to obtain even simpler rule classification systems without a drastic undesirable effect on their accuracy levels; and 2) to analyse the application of IG adaptations to other data mining problems, such as rule association mining [35].

Acknowledgments

This work was supported by the research projects TIN2011-22408 and TIN2012-37930-C02-01.

References

1. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann Publishers, San Francisco, CA, USA (2005)
2. Liao, S.-H., Chu, P.-H., Hisao, P.-Y.: Data mining techniques and applications - A decade review from 2000 to 2011. Expert Syst. Appl. 39(12), 11303–113011 (2012)
3. Richards, D.: Two decades of ripple down rules research. Knowl. Eng. Rev. 24, 159–184 (2009)
4. Cano, A., Zafra, A., Ventura, S.: An interpretable classification rule mining algorithm. Inform. Sciences 240, 1–20 (2013)
5. Cano, J., Herrera, F., Lozano, M.: Evolutionary stratified training set selection for extracting classification rules with trade off precision-interpretability. Data Knowl. Eng. 60, 90–108 (2007)
6. García, S., Fernández, A., Luengo, J., Herrera, F.: A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Comput. 13, 959–977 (2009)
7. Huysmans, J., Dejaeger, K., Mues, C., Vanthienen, J., Baesens, B.: An empiricial evaluation of the comprehensibility of decision table, tree and rule based predictive models. Decis. Support Syst. 51, 141–154 (2011)
8. Culberson, J., Luo, F.: Exploring the k-colorable landscape with iterated greedy. In Cliques, coloring, and satisfiability: Second DIMACS implementation challenge 26, 245–284 (1996)
9. Ruiz, R., Stützle, T.: A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. Eur. J. Oper. Res. 177, 2033–2049 (2007)
10. Lozano, M., Molina, D., García-Martínez, C.: Iterated greedy for the maximum diversity problem. Eur. J. Oper. Res. 214, 31–38 (2010)
11. Rodriguez, F., Lozano, M., Blum, C., García-Martínez, C.: An Iterated greedy algorithm for the large-scale unrelated parallel machines scheduling problem. Comput. Oper. Res. 40(7), 1829–1841 (2013)
12. García-Martínez, C., Rodriguez, F.J., Lozano, M.: Tabu-enhanced iterated greedy algorithm: A case study in the quadratic multiple knapsack problem. Eur. J. Oper. Res. 232, 454–463 (2014)
13. Ying, K.-C., Cheng, H.-M.: Dynamic parallel machine scheduling with sequence-dependent setup times using an iterated greedy heuristic. Expert Syst. Appl. 37(4), 2848–2852 (2010)
14. Lozano, M., Molina, D., García-Martínez, C.: Iterated greedy for the maximum diversity problem. Eur. J. Oper. Res. 214, 31–38 (2011)
15. García-Martínez, C., Rodriguez, F.J., Lozano, M.: Tabu-enhanced iterated greedy algorithm: A case study in the quadratic multiple knapsack problem. Eur. J. Oper. Res. 232, 454–463 (2014)
16. Sokolova, M., Lapalme, G.: A systematic analysis of performance measures for classification tasks. Inform. Process. Manag. 45(4), 427–437 (2009)
17. Ferri, C., Hernández-Orallo, J., Modroiu, R.: An experimental comparison of performance measures for classification. Pattern Recogn. Lett. 30(1), 27–38 (2009)

18. Zafra, A., Ventura, S.: Multi-instance genetic programming for predicting student performance in web based educational environments. Appl. Soft. Comput. 12(8), 2693–2706 (2012)
19. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemannr, P., Witten, I.H.: The WEKA Data Mining Software: An Update. SIGKDD Explorations 11, 10–18 (2009)
20. Bache, K., Lichman, M.: UCI Machine Learning Repository http://archive.ics.uci.edu/ml, University of California, Irvine, School of Information and Computer Sciences (2013)
21. Garcia, S., Molina, D., Lozano, M., Herrera, F.: A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: A case study on the CEC'2005 special session on real parameter optimization. J. Heuristics 15(6), 617–644 (2009)
22. Iman, R., Davenport: J. Approximation of the critical region of the Friedman statistic. In Communications in statistics, 571–595 (1980)
23. Holm, S.: A simple sequentially rejective multiple test procedure. Scand. J. Stat. 6, 65–70 (1979)
24. Wilcoxon, F.: Individual comparisons by ranking methods. Biometrics 1, 80-83 (1945)
25. Bacardit, J., Krasnogor, N.: Performance and efficiency of memetic Pittsburgh learning classifier systems. Evol. Comput. 17, 307–342 (2009)
26. Guan, S., Zhu, F.: An incremental approach to genetic-algorithms-based classification. IEEE T. Syst. Man. Cy. B 35, 227–239 (2005)
27. Tan, K., Yu, Q., Ang, J.: A coevolutionary algorithm for rules discovery in data mining. Int. J. Syst. Sci. 37, 835–864 (2006)
28. González, A., Perez, R.: Selection of relevant features in a fuzzy genetic learning algorithm. IEEE T. Syst. Man. Cy. B 31, 417–425 (2001)
29. Sánchez, L, Couse, I., Corrales, J.: Combining GP operators with SA search to evolve fuzzy rule based classifiers. Inform. Sciences 136, 175–192 (2001)
30. Carvalho, D., Freitas, A.: A hybrid decision tree/genetic algorithm method for data mining. Inform. Sciences 163, 13–35 (2004)
31. Parpinelli, R., Lopes, H., Freitas, A.: Data mining with an ant colony optimization algorithm. IEEE T. Evolut. Comput. 6, 321–332 (2002)
32. Cohen, W.: Fast effective rule induction. Proc. of the 12th International Conference on Machine Learning, 1–10 (1995)
33. Quinlan, J.: C4.5: Programs for Machine Learning (1993)
34. Nauc, D.D.: Measuring interpretability in rule-based classification systems. In Proc. of the IEEE International Conference on Fuzzy Systems, 196–201 (2002)
35. Luna, J.M., Romero, J.R., Ventura, S.: Design and behavior study of a grammar-guided genetic programming algorithm for mining association rules. Knowl. Inf. Syst. 32(1), 53–76 (2012)