Genetic Local Search for Rule Learning

Cristiano Pitangui[1], Gerson Zaverucha[1]
COPPE – PESC/UFRJ - Rio de Janeiro, Brasil.
{cris_pi, gerson}@cos.ufrj.br

ABSTRACT
The performance of Evolutionary Algorithms for combinatorial problems can be significantly improved by adding Local Search, thus obtaining a Genetic Local Search (GLS) also called Memetic Algorithm. In this work, we adapt a previous Stochastic Local Search (SLS) algorithm and embed it into a GBML system. The adapted SLS algorithm works as a module of the system that tries to improve a random individual in the population. We perform experiments to evaluate this adapted SLS procedure and results show that this new GLS system is very effective, not losing in any of the 10 UCI datasets tested when compared to the system without the SLS procedure. The system either obtained significantly more accurate concepts using lower number of rules and features or it achieved the same accuracy as the system without the SLS procedure, but reduced the number of rules and features, and also the time taken to develop the solution.

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

General Terms: Algorithms, Performance, Experimentation.

1. INTRODUCTION
Genetic Based Machine Learning (GBML) states for the application of Genetic Algorithms (GA) [1] to Machine Learning. Pure evolutionary algorithms often seem to lack the capability of sufficient search intensification, that is, the ability to reach high-quality candidate solutions efficiently after the application of the genetic operators (crossover, mutation and selection). Hence the performance of Evolutionary Algorithms for combinatorial problems can be significantly improved by adding Local Search [2]. A Genetic Local Search (GLS) can be defined as a Genetic Algorithm embedded with a Local Search technique [2].

One possible way of alleviating the huge requirements of search in such large spaces is to take advantage of clever local search strategies as Stochastic Local Search (SLS). In [3], Rückert and Kramer proposed a SLS algorithm for learning k-term DNFs, that is, a set of k rules, obtaining competitive experimental results compared to state-of-the-art rule learners. Therefore, we would like to embed Rückert and Kramer SLS algorithm, as the local search phase of a GBML rule learner.

In [4, 11] we have developed a GBML rule learner that uses the very common Modified DNF representation [5] and the high memory economy Natural Coding [6]. Also, it was proposed a new crossover operator for discrete attributes that explores the search space like the two points binary crossover. This way, the system combines the memory economy given by the Natural Coding with the same exploration power of the two points binary crossover. Moreover, the system merges the Michigan and Pittsburgh rule generation approaches and also uses the Implicit

Feature Selection Mechanism (IFSM) that tries to reduce the number of features used in a rule. It achieved very good experimental results compared with C4.5 [7].

The paper is structured as follows. Section 2 describes the main aspects of SLS algorithms and presents the new GLS rule learner. The experimental results are shown in section 3. Finally, section 4 concludes the work.

1- The first author is financially supported by the Brazilian Research Agency CAPES. The second author is partially financially supported by the Brazilian Research Agency CNPq

GECCO'08, July 12–16, 2008, Atlanta, Georgia, USA.
ACM 978-1-60558-130-9/08/07.

2. STOCHASTIC LOCAL SEARCH
SLS algorithms perform a local randomized-walk in the search space [2]. Generally, the SLS starts with a randomly generated solution and it is usually organized in two main steps: Search and Selection.

In the Search step, the algorithm examines a set of neighbors (candidate solutions) according to a neighbourhood relation. Each neighbor is evaluated by a global scoring function.

In the Selection step, the algorithm selects the candidate with highest score given by the global scoring function. Since the choice of the best candidate can easily lead to local optima, the algorithm with a probability p chooses a random candidate solution instead of the best one. (See [3] for a detailed description of a typical SLS algorithm).

2. 1 Embedding SLS in our GBML System
Basically, our hybrid approach to develop concepts has a Michigan cycle inside a Pittsburgh cycle. This way, we call the SLS procedure inside the Pittsburgh cycle, after Michigan cycle execution (see algorithm 1).

When the GLS system calls the SLS procedure, one individual of

Algorithm 1. GLS Algorithm.

- **Input**: **maxn**: maximal number of generations; **k, j**: natural numbers that, respectively, determine when the Michigan cycle and SLS occur; **pp**: principal population (for Pittsburgh cycle); **sp**: sub-evolution population (for Michigan cycle); **e+**: the set of all positive examples in dataset; **e-**: the set of all negative examples in dataset; **enc+**: the set of positive examples not covered by the best individual in pp;
- **Output**: modified pp.;

Begin:

```
1 - While (generation < maxn)                    //Pittsburgh Cycle
  2 - Evolve pp with e+ ∪ e-;
  3 - If ( (generation  % k) == 0 ) then         //Michigan Cycle
      4 – Initialize sp randomly;
      5 - Evolve sp with enc+ ∪ e-;
      6 - pp ← pp concatenated with sp;          //End of Michigan cycle
  7 - If ( (generation  % j) == 0 ) then
      8 - a random S from pp ← SLS (a random S, p1, p2, p3);
9 – Return pp;                                   //End of Pittsburgh cycle
```

the population is randomly selected to go through the SLS. After the execution of the SLS procedure, this improved individual is placed again in the population to continue its genetic evolution through the Pittsburgh cycle.

The algorithm 2, called as SLS at algorithm 1, shows how SLS algorithm from [3] is adapted for our GBML.

Algorithm 2. Rückert and Kramer SLS adapted for our GBML

- **Input:** solution (**s**) with one or more rules; probabilities (**p1, p2, p3**); **maxScore; maxSteps**.
- **Output:** Modified solution (**s**).

Begin:

1- While fitness (s) ≠ maxScore or steps < maxSteps do:
  2- ex ← a random example misclassified by **s**;
  3- If ex is a **positive** example then
    4- rule ← with a probability p1: a random rule from the solution **s**; otherwise, the rule that differs in the smallest number of attributes from ex;
    5- atr ← with a probability p2: a random attribute in rule; otherwise, the attribute that when is not used, increases the fitness from solution **s** at most;
    6- **s** ← **s** with the atr removed (not used) from rule;
    end-if;
  7- If ex is a **negative** example then
    8- rule ← a random rule in solution **s** that covers ex;
    9- atr ← with a probability p3 a random attribute from **s**, that when is being used in rule, this rule does not cover ex; otherwise, an attribute from **s** that when is being used in rule, increases the fitness from solution **s** at most;
    10- **s** ← **s** with attribute added (used) to rule;
  end-if;
end-while;
11- return solution **s**:

3. EXPERIMENTS

This section presents the results obtained by the GA without the SLS procedure compared to the new GLS. All the experimental results were obtained using 5 fold stratified cross validation, where in each fold an internal 5 fold stratified cross validation was used for tuning the parameters for all systems. For continuous-valued attributes, we used the discretization method proposed in [9]. For all experimentations, corrected-Student two-tailed paired t-tests [10], using a confidence interval of 95%, were used in order to analyze the results. The fitness function, for both GAs, was the training accuracy. For each fold, the algorithm was executed 20 times obtaining an average for the fold; as usual, the results are the average (on the test sets) of the 5-fold stratified cross validation.

Table 1. Results: GA without SLS and GLS

	GA without SLS			**GLS**			
Nam.	#Ru	#Att	Acc	#Rul	#Att	Acc	T-Test
bre	1	2,4	76.1	1	2,4	76,1	no
cr-a	1.4	3.2	92.4	1.4	3.2	92.4	no
gls	9	27.8	78.5	6,6	16	78,5	no
he-c	2.6	6.4	**86.8**	2,4	6,6	**88,5**	yes
hep	1.8	3.8	88.4	1.2	2.8	88.4	no
h-col	1.8	5	**84.8**	1,6	4,6	**88,1**	yes
Ino	2.4	19.8	**91.3**	1,2	13,2	**95,6**	yes
son	1.6	5	**79.0**	1,6	4	**83,0**	yes
vehi	14	82.6	**74.0**	10	70,8	**77,3**	yes
wbcd	2	4.2	96.1	1,6	3,2	97,7	no

The table 1 shows that the accuracy obtained by the new GLS is better or equal to the accuracy obtained by GA without SLS. In 5 of 10 datasets, the system using SLS achieved significantly better accuracy and simpler concepts. For hep and glass, the accuracy was the same obtained with GA without SLS, but the concepts were simpler. It is important to note that, for the datasets that the GLS did not achieved better accuracies nor a lower number of rules and features (bre and cr-a), these results were obtained in less time (lower number of population size, generations and sub-generations). In fact, in comparison to the GA without SLS, the GLS uses an inferior number of population size, generations and sub-generations. These numbers are respectively 33%, 25% and 33% lower.

4. CONCLUSION

This paper presents an adaptation of Rückert and Kramer SLS algorithm [3] for our previous GBML system [4, 11].

Results show that this procedure can help our GBML system to evolve simpler and more accurate concepts using less time than without the SLS procedure. This new GLS system showed to be very effective, not losing in any of the 10 datasets tested and significantly improving the accuracy and the simplicity in 5 of them. Even when it was as accurate as GA without SLS, the GLS developed simpler concepts or it was significantly faster.

5. REFERENCES

[1] David. E. Goldberg., *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Massachusetts, 1989.

[2] H. Hoos, T. Stützle*, Stochastic Local Search: Foundations and Applications*, Morgan Kaufmann, The Morgan Kaufmann Series in Artificial Intelligence, Inc. San Francisco, CA, USA 2004.

[3] Ulrich Rückert and Stefan Kramer. "Stochastic local search in k-term DNF learning." *In Proc. Of the 20th ICML*, pages 648-655, 2003.

[4] Pitangui, C., Zaverucha, G. "Genetic Based Machine Learning: Merging Pittsburgh and Michigan, an Implicit Feature Selection Mechanism and a New Crossover Operator". *6th International Conference on Hybrid Intelligent Systems.* Auckland, New Zealand, 2006.

[5] K. A. DeJong, W. M. Spears, and D. F. Gordon, "*Using genetic algorithms for concept learning*", Machine Learning, vol. 1, no. 13, 1993, pp. 161-188.

[6] Jesús S. Aguilar-Ruiz and J.C. Riquelme. "Improving the Evolutionary Coding for Machine Learning Tasks". *European Conference on Artificial Intelligence*, IOS Press, Lyon, France 2002, pp 173-177

[7] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Francisco, 1993.

[8] Blake, C.L. and Merz, C.J., "UCI Repository of machine learning databases*",* Irvine, CA: University of California, Department of Information and Computer Science, 1998. (http://archive.ics.uci.edu/ml/)

[9] U. M. Fayyad and K. B. Irani. "Multi-interval discretization of continuous valued attributes for classification learning", *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, Chambery, France, 1993, pp. 1022-1027.

[10] Nadeau C., Bengio Y., "*Inference for the Generalization Error*", Machine Learning 52(3) pp. 239-281, 2003.

[11] Pitangui, C., Zaverucha, G. Improved Natural Crossover Operators in GBML. In: IEEE Congress on Evolutionary Computation (CEC), 2007, Singapore, 2007. p. 2157-2164.