

LEMMO: Learnable Evolution Model for Multi-Objective

L. Jourdan^{1,2}, D. Corne¹, D. Salvic² and G. Walters²

¹ Department of Computer Science, Harrison Building,
University of Exeter,
Exeter EX4 4QF,
United Kingdom

`jourdan@lifl.fr`, `D.W.Corne@ex.ac.uk`

² Department of Water Systems, Harrison Building,
University of Exeter,
Exeter EX4 4QF,
United Kingdom

`D.Savic@exeter.ac.uk`, `G.A.Walters@ex.ac.uk`

Abstract. In this article, we present for the first time LEMMO: Learnable Evolution Model for Multi-Objective that is designed to reduce the number of evaluations in cost evaluation problems and to boost the robustness of the multi-objective genetic algorithm. LEMMO is here experiment on a real water distribution optimization problem in order to show first obtained results. On this problem, LEMMO has shown that it can be a promising solution for faster optimization needs.

1 Introduction

Fast optimization is more and more essential when facing to real problems that are more and more time consuming for evaluating a potential solution. There is a compromise to obtain between only speeding up the optimization method and keeping good quality in obtained solutions (see figure 1).

In this article, we propose an evolutionary multi-objective method that is based on machine learning in order to lead search to promising area through the generations. In the next part, we describe the relative works and in particular LEM that has conduct us to develop LEMMO. Then, we present the application that has been used to test our method. Finally we give and discuss results on different implemented schemes on a water system problem.

2 LEMMO

There are several works that introduce machine learning in evolutionary algorithms as the works of Ravise and Sebag [16, 14, 15] or Handa [2–4]. These works are only on bit representation and for mono-objective problems. A more general approach can be found in the work of Michalski, LEM.

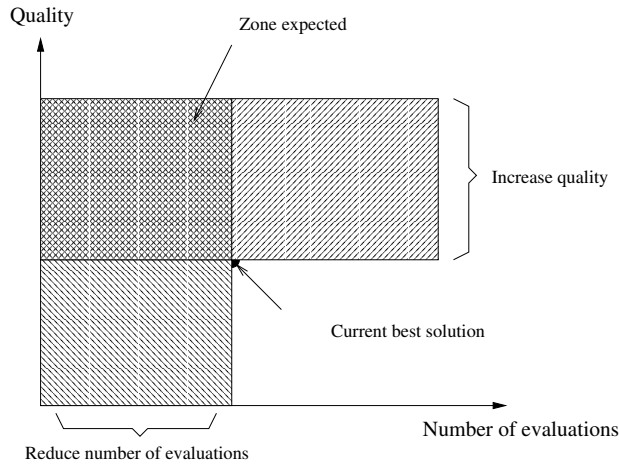


Fig. 1. Partition of the potential solutions.

2.1 LEM

The Learnable Evolution Model (LEM) [8, 9] integrates a symbolic learning component to evolutionary computation; it seeks out rules explaining the differences between the better and worse performers in the population, and generates new individuals based on the templates specified in these rules. The LEM methodology has proved able to improve on the efficiency of the evolutionary process. LEM uses AQ learning algorithm in order to produce rules. Existing implementations, such as AQ11 [10], AQ15 [11] handle noise with pre and post-processing techniques. The basic AQ algorithm however, heavily depends on specific training examples during search (the algorithm actually employs a beam search).

2.2 Adaptation to Multi-Objective genetic algorithm

LEMMO will seek for rules that explain why some individuals dominate others in a multi-objective point of view (class good) and why some individuals are dominated by others (class bad). It will create new individuals thanks to the rules by creating solutions that match to positive rules and don't match to negative rules. We will see that there are several ways to apply such a mechanism and that some are better than others.

LEMMO is designed to take any rule induction algorithm in order to produce rules to create new individuals.

In this first implementation, we have decided to use C4.5 as a rule induction algorithm. The best known algorithm C4.5 takes a set of examples as input and produces a decision tree which is consistent with examples [12]. From the tree, it is easy to obtain rules that suit examples with C4.5rules which is furnished with C4.5 by Quinlan. An example of rules generated in our application is given figure 2.

In order to use such a method, we have to face to problem: how to define the sample set for C4.5 and how to construct the new solutions from the generated rules.

Fig. 2. Example of produced rules by C4.5 and C4.5rules for continuous attributes.

```
R1: A3 <= 0 AND A7 <= 6 AND A18 <= 4
-> class bad
R2: A19 <= 4 AND A21 <= 1
-> class bad
R3: A7 > 6 AND A10 <=5
-> class good
R4: A18 > 4
-> class good
```

Construction of the sample set for the inducer algorithm In order to produce rules, C4.5 need a set of examples to construct the decision tree. In a mono-objective problem, Michalski [8,9] took the 30% best individuals and the 30% badest over the population. In a multi-objective point of view, the construction of the sample file is more difficult. There is a lot of possibility for constructing the set and we will test some of them.

Generation of new individuals We generate several new individuals for each rule that is of the class good (called also positive rule). The new individuals will suit one of the positive rules in his genotype. For example, the individual *****9**3***** will suit the rule R3. Only a few part of the genotype is decided by the rule, the rest of the genotype is for the moment randomly generated as the authors shown that even randomly completion can give good results. Then we verify that the new generated individuals do not match with negative rules and correct the individual in matching case. All the process is recall in figure 3.

3 Application to water System

Water distribution network modeling is used for a variety of purposes such as: strategic and master planning; system design; energy management and daily operations; fire protection studies and emergency responses; water quality investigations; and many others. The computer modeling of water distribution networks continues apace in the water industry as computers become increasingly powerful, more complex systems are available to be modeled. Open source

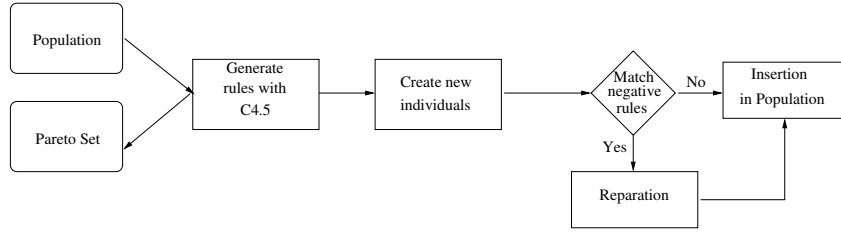


Fig. 3. Creation of new individuals.

software such as EPANET [13]. In recent years, computational methods such as non-linear programming, dynamic programming and search techniques have been used to optimize the design, operation and rehabilitation of these networks.

The optimization of WDN models can primarily be used for three purposes: the design of networks for new supply areas (design problems); modifying existing designs to meet new demands or other factors (rehabilitation problems) and modifying network parameters to ensure that they are accurate with respect to the real world (calibration problems). This paper is concerned with the problems of rehabilitation and design although the methods described herein can theoretically be used for any of these problems. A standard rehabilitation problem is concerned with taking an existing network and modifying its parameters or components to satisfy new constraints in the performance of the network. Therefore any process designed to optimize a rehabilitation problem must consider both the performance of the network (to be maximized) and the cost of the proposed solution (to be minimized). A design problem is similar except that instead of deciding on pipe sizes to meet some new criteria, the pipe sizes of the entire network must be decided from scratch.

3.1 NY problem

The New York pipe network has been studied using evolutionary techniques. The objective of the NY problem was to determine the most economically effective design for addition to existing system of tunnels that constituted the primary water distribution system of the city of New York. Tunnel (pipe) diameters are considered as design variables. There are 15 available discrete diameters (36, 48, 60, 72, 84, 96, 108, 120, 132, 144, 156, 168, 180, 192, 204 inches) and one extra possible decision which is 'do nothing' option. The minimum head requirement at all nodes is fixed at 255 ft except for node 16, 17 and 1 that are 260, 272.8 and 300 ft respectively. All the twenty one tunnels are considered for duplication. Supplying demand at an adequate pressure to consumers is the main constraint in the design of water distribution systems. Each pipe as a cost. The cost function is a non-linear function:

$$C = 1.1 \times D_{ij}^{1.24} \times L_{ij}$$

in which cost C is in dollars, diameter D_{ij} is in inches, and length L_{ij} is in feet. A full enumeration of all possibilities would require: $16^{21} = 1.9342 \times 10^{25}$.

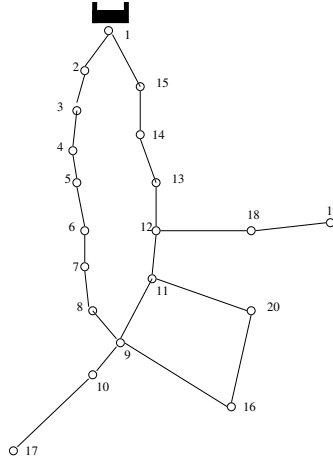


Fig. 4. The New York City problem.

From an optimization perspective, the objective of the NY problem is to modify the rehabilitated pipe diameters to meet the demands at the nodes. The current optimal solution for this is 38.64 million dollars and no pressure deficit although this can vary slightly depending on the modeling software and parameters used.

3.2 Fitness landscape study

As suggested in [7], we decided to study the fitness landscape of the NY problem by executing 1000 multi-objective Hill-Climbing algorithms from random points. The objective is to determine how are spread the solutions from the best one obtained by the 1000 hill-Climbing noted α . In this study, we decided to measure the distance between the individuals and α by counting the numbers of pipes that are not similar to the pipes of α .

The figures 5 and 6 illustrate the obtained repartition for both objectives: cost and head deficit. Such an experiment shows that the landscape is very large and not easy for local search methods. Evolutionary algorithms are well adapted to such problems.

4 Experiments

The genetic algorithm uses NSGA-II [1]. One individual contains the chosen option for the pipe. The crossover used is onepoint and the mutation is a random mutation. The population size is 100, the crossover rate is 0.9 and the mutation rate is 0.9.

The different tested schemes are:

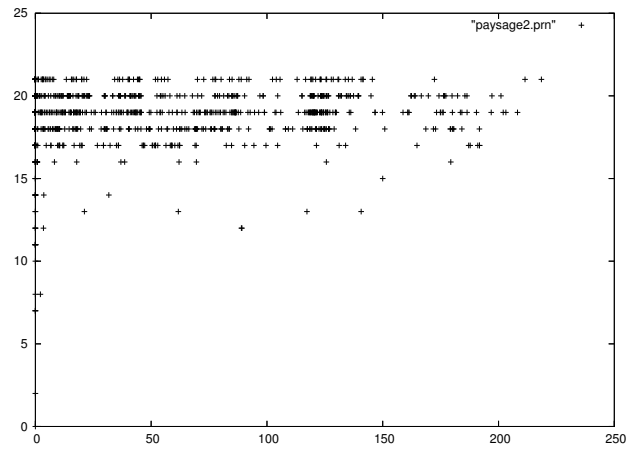


Fig. 5. Repartition of the solutions of the Hill-Climbing for the head deficit.

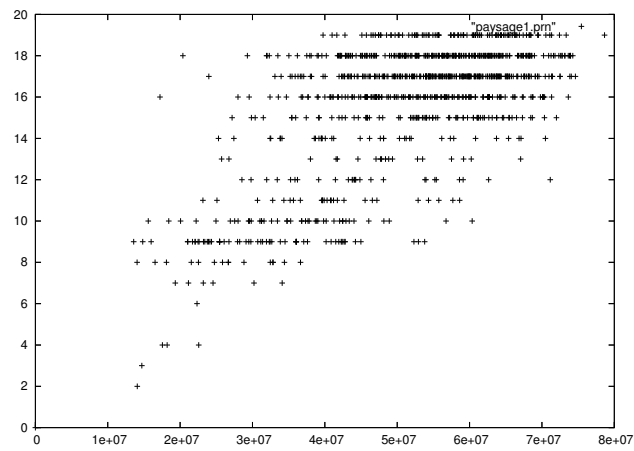


Fig. 6. Repartition of the solutions of the Hill-Climbing for the cost.

- LEMMO-1: run C4.5 crossover when there is no improvement of the Pareto front
- LEMMO-fix1: run C4.5 crossover every 10 generations, the rules are generated from the first inserted individuals of the Pareto set and the individuals of the current population
- LEMMO-fix2: run C4.5 crossover every 10 generations, the rules are generated from the last inserted individuals of the Pareto set and the individuals of the current population
- LEMMO-fix3: run C4.5 crossover every 10 generations, the rules are generated from the randomly chosen individuals of the Pareto set and the individuals of the current population
- LEMMO-fix4: run C4.5 crossover every 10 generations, the rules are from the best individuals for one objective. The objectives are treated alternatively. This individual are taken from the Pareto set and the current population.

4.1 Quality measures

As suggested in [6], we use several metrics in order to assess the quality of our method.

S metric

A definition of the S metric is given by Zitzler in [17]. Let A be a non-dominated set of solutions. S metric calculates the hyper-volume of the multi-dimensional region enclosed by A and a reference point Z_{ref} .

We take for this problem as Z_{ref} the worst possible solution. This solution determines the bound of the space search: for the cost, we take for all pipe the largest diameter (this solution is the most expensive), for the head deficit we take for all pipe the minimum diameter (this solution as no pressure for all clients). In order to compare evolution of each algorithm, we have normalized the S-metric such as the maximal theoretical area is 1.0.

$R1_R$ Metric

The $R1$ metric [5] is based on calculating the probability that the approximation A is better than B over an entire set of utility functions. $R1_R$ is $R1$ when it is used with a reference set i.e. as a reference metric. This metric does then induce a total ordering on the set of approximation.

$$R1(A, B, U, p) = \int_{u \in U} C(A, B, u) p(u) du$$

with U a set of utility functions A and B two Pareto sets.

$$C(A, B, u) = \begin{cases} 1 & \text{if } u^*(A) > u^*(B) \\ 1/2 & \text{if } u^*(A) = u^*(B) \\ 0 & \text{if } u^*(A) < u^*(B) \end{cases}$$

We can notice that $R1(A, B, u, p) = 1 - R1(B, A, u, p)$. If $R1(A, B, u, p) > 1/2$ then A is better than B .

4.2 Results on NY problem

Each scheme has been tested 30 times. The obtained Pareto fronts have several non-dominated solutions (see figure 7). Our objective is to discover which scheme can be the more interesting and if the parameter of the scheme is the good one. We will also use boxplots to present a graphical view of the results. The boxplot provides a method for graphically depicting the distribution of a dataset and comparing with others. The median for each dataset is indicated by the black center line, and the first and third quartiles are the edges of the area, which is known as the inter-quartile range (IQR). The extreme values (within 1.5 times the inter-quartile range from the upper or lower quartile) are the ends of the lines extending from the IQR. Points at a greater distance from the median than 1.5 times the IQR are plotted individually as circle and '+'. These points represent potential outliers.

Quality

The table 1 gives the statistics on the quality of the S metric of the tested scheme compared to NSGA-II.

We have observed through the experiments that introducing machine learning in the multi-objective genetic algorithm increase the quality of the results for the S-metric for any tested scheme. The robustness of the algorithm is also increased as the Standard deviation is decrease by more than 50% by two schemes (LEMMO-fix1 and LEMMO-fix3). The boxplot figure 8 shows that the most reliable schemes for Smetric is LEMMO-fix3 et LEMMO-fix4.

For the $R1_R$ metric (see table 2), we compare all the scheme to one front of NSGA-II. The objective is to have the minimal $R1_R$ value. We observe that all the schemes are better than NSGA-II for this metric. The boxplot in figure 9 shows that for $R1_R$ the most reliable scheme are also LEMMO-fix3 et LEMMO-fix4.

We have also made a random statistical test on all the $R1_R$ results. We have observed the the scheme LEMMO-fix4 is the most significant.

Numbers of evaluation

Table 3 present the variation of the number of evaluations required to obtained different values of the normalized S metric between the scheme and NSGA-II over 30 runs. The first value 0.955 is chosen for this problem as it show the beginning of the convergence of the algorithm for this problem. The second value 0.968 is chosen as it is rather at the end of the convergence.

All the schemes are faster than NSGA-II to obtained a normalized Smetric greater than 0.968. We notice that LEMMO-fix4 is the less expensive scheme in comparison of NSGAI in median (-16.66%) and in mean (-20.58%) to obtain this value.

If we have a look to each criterion of our comparison of possibility for LEMMO, we can notice that LEMMO-fix4 seems to be the most interesting. The figure 10 illustrates for the scheme LEMMO-fix4 the gap with NSGA-II over one execu-

tion.

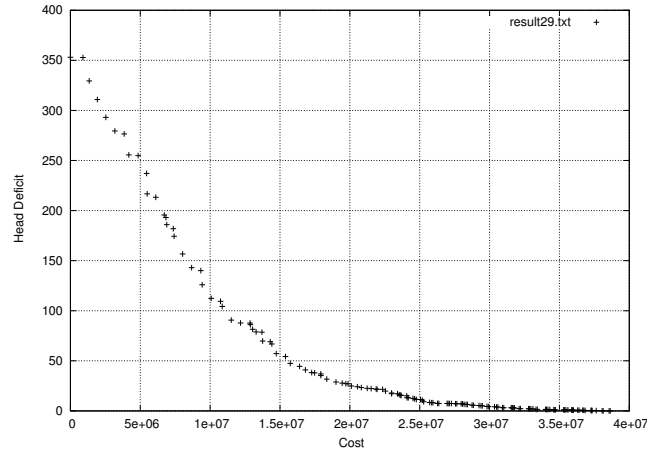


Fig. 7. Example of obtained Pareto front on NY problem.

Table 1. Quality assessment S metric ($(S(\text{Scheme}) - S(\text{NSGA-II}))/S(\text{Scheme})$) on NY problem.

Scheme	S Median	S Mean	S Min	S Max	S Std Dev.
LEMMO-1	7.83%	4.9%	7.3%	0.26%	-33.32%
LEMMO-fix1	8.28%	5.44%	3.4%	0.26%	-57.6%
LEMMO-fix2	8.19%	5.5%	2.04%	0.18%	-25.9%
LEMMO-fix3	8.14%	6.75%	8.84%	0.42%	-58.58%
LEMMO-fix4	9.12%	6.8%	7.77%	0.27%	-22.19%

5 Conclusion

In this article we have presented LEMMO and tested on a classical water system problem different option of implementation of LEMMO according to two

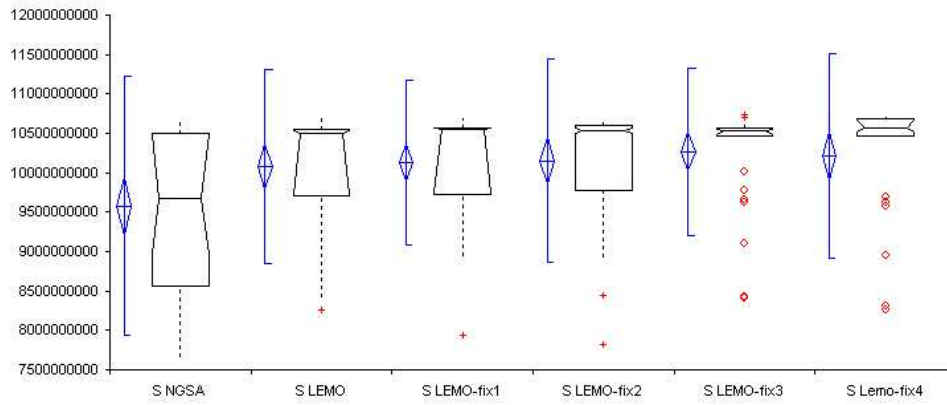


Fig. 8. Boxplot of S .

Table 2. Quality assessment $R1_R$ metric with a front of NSGA-II for NY problem.

Scheme	$R1_R$ Median	$R1_R$ Mean	$R1_R$ min	$R1_R$ max	$R1_R$ Std Dev.
NSGA-II	0.39172	0.44780	0.37525	0.62271	0.09460
LEMMO-1	0.38323	0.43119	0.35229	0.64671	0.09236
LEMMO-fix1	0.38024	0.41153	0.35229	0.64271	0.09044
LEMMO-fix2	0.38025	0.39476	0.35229	0.51597	0.05348
LEMMO-fix3	0.38124	0.40108	0.35229	0.65469	0.07496
LEMMO-fix4	0.38224	0.39680	0.36527	0.51497	0.04350

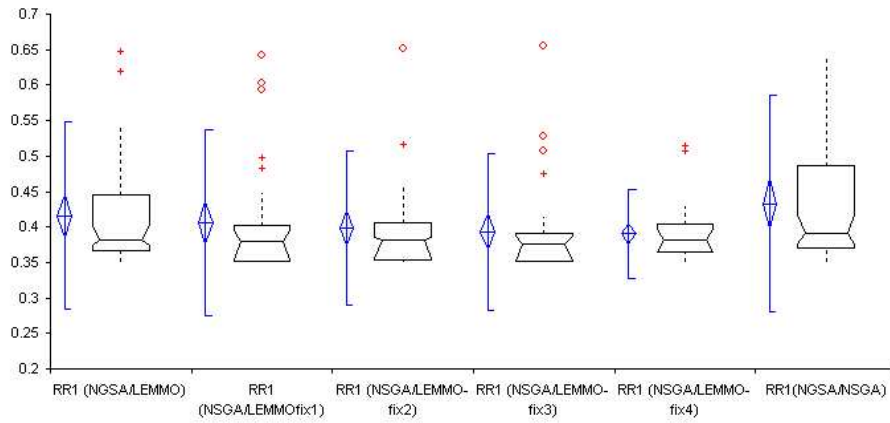


Fig. 9. Boxplot of $R1_R$.

Table 3. Statistic on number of evaluation for finding a normalized Smetric greater than 0.955 and 0.968 over 30 runs for the different tested schemes (T(Scheme)-T(NSGA-II)/T(Scheme)) on NY problem.

Scheme	$S_{norm} > 0.955$		$S_{norm} > 0.968$	
	Median	Mean	Median	Mean
LEMMO-1	0%	+16.16%	-16%	-6.8%
LEMMO-fix1	+10%	+10%	-10.10%	-5.01%
LEMMO-fix2	+16.66%	+16.66%	-7.8%	+2.6%
LEMMO-fix3	+6.25%	+14.28%	-13.51%	-10.81%
LEMMO-fix4	+10 %	0%	-16.66 %	-20.58%

Table 4. Impact of the parameters: $R1_R$ metric with a front of NSGA-II on NY problem.

Scheme (parameter)	$R1_R$ Median	$R1_R$ Mean	$R1_R$ Std Dev.
LEMMO-fix1 (5)	0.38124	0.38503	0.02553
LEMMO-fix1 (10)	0.38024	0.41152	0.09044
LEMMO-fix1 (20)	0.38024	0.42664	0.12099
LEMMO-fix4 (5)	0.40118	0.47329	0.07234
LEMMO-fix4 (10)	0.38224	0.39681	0.0435
LEMMO-fix4 (20)	0.37924	0.39654	0.07272

Table 5. Statistic on number of evaluation for finding a normalized Smetric greater than 0.955 and 0.968 over 30 runs for the different tested schemes (T(Scheme)-T(NSGA-II)/T(Scheme)) on NY problem.

Scheme (parameter)	$S_{norm} > 0.955$		$S_{norm} > 0.968$	
	Median	Mean	Median	Mean
LEMMO-fix1 (5)	+21.05%	+21.21%	+2.31	+4.65%
LEMMO-fix1 (10)	+10%	+10%	-10.10%	-5.01%
LEMMO-fix1 (20)	+16.66%	+16.66%	+13.51%	+2.38%
LEMMO-fix4 (5)	+6.25%	+16.66%	-14.28%	-7.89%
LEMMO-fix4 (10)	+10 %	0%	-16.66 %	-20.58%
LEMMO-fix4 (20)	+6.89%	+6.66%	-2.43%	+2.43%

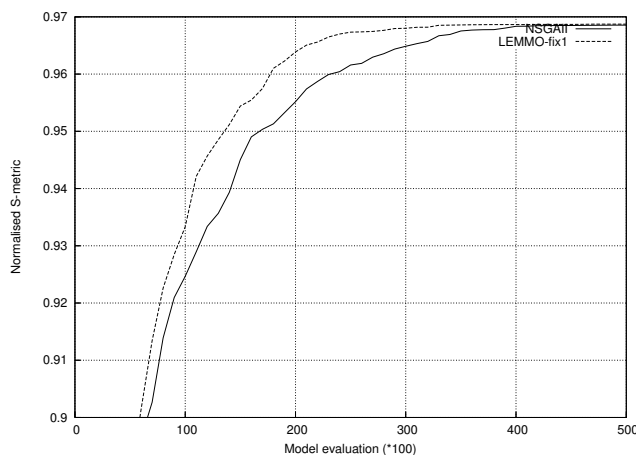


Fig. 10. Example of evolution of the normalized S-metric during the evolution against model evaluation on NY problem.

quality measures and the number of evaluation needed to reach different values of normalized Smetric.

LEMMO has shown that introducing machine learning can be a good way to both decreased the numbers of evaluation and increased the quality of the obtained Pareto front. More experiments have to be conduct in order to see the behavior of LEMMO on larger problems.

The future works will be to compare different machine learning algorithms: other rule induction methods, classical data mining algorithm such as Apriori and different completions of generated rules.

LEMMO can be applied to many other real problem where the evaluation is time consuming.

References

1. K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-II. In *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pages 849–858, 2000.
2. H. Handa, T. Horiuchi, O. Katai, and M. Baba. A novel hybrid framework of coevolutionary ga and machine learning. *International Journal of Computational Intelligence and Applications*, 2002.
3. H. Handa, T. Horiuchi, O. Katai, T. Kaneko, T. Konishi, and M. Baba. Co-evolutionary ga with schema extraction by machine learning techniques and its application to knapsack problems. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pages 1213–1219, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 May 2001. IEEE Press.
4. H. Handa, T. Horiuchi, O. Katai, T. Kaneko, T. Konishi, and M. Baba. Fusion of coevolutionary ga and machine learning techniques through effective schema

- extraction. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, page 764, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.
5. M. P. Hansen. *Metaheuristics for multiple objective combinatorial optimization*. PhD thesis, of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, Report IMM-PHD-1998-45, 1998.
 6. J. D. Knowles and D. W. Corne. On metrics for comparing non-dominated sets. In IEEE Service Center, editor, *Congress on Evolutionary Computation (CEC'2002)*, volume 1, pages 711–716, Piscataway, New Jersey, May 2002.
 7. P. Merz. *Memetic Algorithms for Combinatorial Optimization Problems: Fitness Landscapes and Effective Search Strategies*. PhD thesis, Department of Electrical Engineering and Computer Science, University of Siegen, Germany, 2000.
 8. R.S. Michalski. Learnable evolution model: Evolutionary processes guided by machine learning. *Machine Learning*, 38(1–2):9–40, 2000.
 9. R.S. Michalski, G. Cervon, and K.A. Kaufman. Speeding up evolution through learning: Lem. In *Intelligent Information Systems 2000*, pages 243–256, 2000.
 10. R.S. Michalski and J.B. Larson. Selection of most representative training examples and incremental generation of v11 hypothesis: The underlying methodology and the descriptions of programs esel and aq11. Technical Report Report No. 867, Urbana, Illinois: Department of Computer Science, University of Illinois, 1978.
 11. R.S. Michalski, I. Mozetic, J. Hong, and N. N. Lavrac. The multipurpose incremental learning system aq15 and its testing application to three medical domains. In *Proc. of the Fifth National Conference on Artificial Intelligence*, pages 1041–1045. PA: Morgan Kaufmann, 1986.
 12. J. Ross Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., 1993.
 13. L.A. Rossman. Epanet, users manual. Technical report, U.S. Envi. Protection Agency, Cincinnati, Ohio, 1993.
 14. M. Sebag, C. Ravise, and M. Schoenauer. Controlling evolution by means of machine learning. In *Evolutionary Programming*, pages 57–66, 1996.
 15. M. Sebag and M. Schoenauer. Controlling crossover through inductive learning. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Parallel Problem Solving from Nature – PPSN III*, pages 209–218, Berlin, 1994. Springer.
 16. M. Sebag, M. Schoenauer, and C. Ravise. Toward civilized evolution: Developing inhibitions. In Thomas Bäck, editor, *Proc. of the Seventh Int. Conf. on Genetic Algorithms*, pages 291–298, San Francisco, CA, 1997. Morgan Kaufmann.
 17. E. Zitzler. Evolutionary algorithms for multiobjective optimization: Methods and applications. Master's thesis, Swiss federal Institute of technology (ETH), Zurich, Switzerland, november 1999.