Preliminary Investigation of the 'Learnable Evolution Model' for Faster/Better Multiobjective Water Systems Design

Laetitia Jourdan, David Corne, Dragan Savic, and Godfrey Walters

Schoolof Engineering, Computer Science and Mathematics, Harrison Building, University of Exeter,Exeter EX4 4QF, United Kingdom
`jourdan@lifl.fr`, {`D.W.Corne, D.A.Savic, G.A.Walters`}`@ex.ac.uk`

Abstract. The design of large scale water distribution systems is a very difficult optimisation problem which invariably requires the use of time-expensive simulations within the fitness function. The need to accelerate optimisation for such problems has not so far been seriously tackled. However, this is a very important issue, since as MOEAs become more and more recognised as the 'industry standard' technique for water system design, the demands placed on such systems (larger and larger water networks) will quickly meet with problems of scaleup. Meanwhile, LEM (Learnable Evolution Model') has appeared in the Machine Learning literature, and provides a general approach to integrating machine learning into evolutionary search. Published results using LEM show very great promise in terms of finding near-optimal solutions with significantly reduced numbers of evaluations. Here we introduce LEMMO (Learnable Evolution Model for Multi-Objective optimization), which is a multi-objective adaptation of LEM, and we apply it to certain problems commonly used as benchmarks in the water systems community. Compared with NSGA-II, we find that LEMMO both significantly improves performance, and significantly reduces the number of evaluations needed to reach a given target. We conclude that the general approach used in LEMMO is a promising direction for meeting the scale-up challenges in multiobjective water system design.

1 Introduction

Fast optimization (in terms of using as few fitness evaluations as possible) is more and more essential when faced with real-world problems in which each fitness evaluation involves running an time-expensive simulation. However, there is of course a compromise between speeding up the optimization method, and ensuring good quality in the obtained solutions (see figure 1). One area in which this issue is paramount is in the design of large scale water distribution networks. This area contains various difficult and complex optimisation problems which invariably requires the use of time-expensive simulations within the fitness function. These problems are typically multi-objective (often trading off financial cost against requirements for pressure, speed of flow and other aspects of the

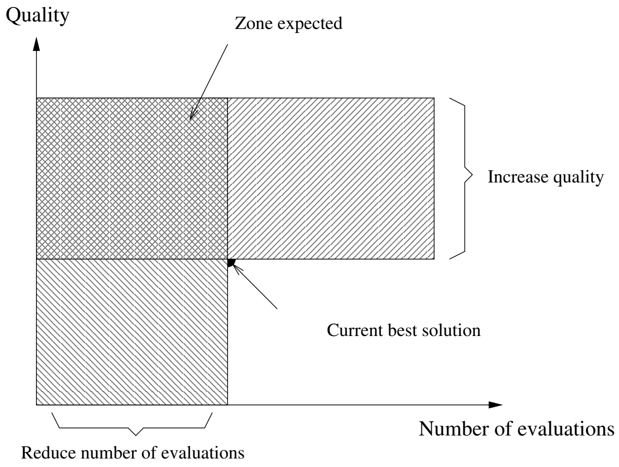

Fig. 1. Partition of the potential solutions

required design), and there is an increasing body of published research which addresses such problems, e.g.: [2, 26, 9, 23, 6, 4, 7, 1] However, the need to accelerate optimisation for such problems has not so far been seriously tackled. This is presumably because the problems involved are those of *design*, and sufficient time often exists to allow long optimisation runs before a final design is to be scrutinised and approved. However, as the quality of MOEA approaches causes them to be considered for ever larger problems, the time taken to find good solutions to such larger problems (e.g. a water design network with thousands of pumps, which is not uncommon) may turn out to be unacceptable with existing methods.

In this article, we address this scale-up problem by investigating a method based on the Learnable Evolution Model (LEM) [16, 17], which has appeared in recent years in the machine learning literature, but has so far been very little explored on real problems. The idea, which involves combining machine learning with evolutionary search, is a very generalised notion of which certain current trends in evolutionary computation (such as Estimation of Distribution Algorithms [15]) can be seen as specific instances. In particular, results to date on benchmark function optimisation problems show that LEM can save very significantly on the number of fitness evaluations needed to reach a certain target fitness. Our adaptation of this method to multiobjective optimisation, and in particular its application to water network design problems, is investigated here.

The remainder of the article is set out as follows. In section 2 we describe LEM and LEMMO (our version of this method). In section 3 we describe water systems design optimisation problems in general, and present the benchmark problems we use and briefly describe the associated fitness function simulator. In section 4, we describe the implementation of our genetic algorithm, the different variants of LEMMO and the metrics we use. Section 5 presents results from our LEMMO method, compared against NSGA-II (a well-known high-quality MOEA). Some concluding discussion is provided in section 6.

2 LEMMO

There are several publications that investigate the hybridisation of machine learning and evolutionary algorithms, e.g. Sebag [29, 27, 28, 10, 11, 12], while some works concentrate on statistical learning integrated within an EA [21, 15]. These and similar studies so far have concentrated on binary encodings and have rarely strayed from single-objective optimisation. However, a recent highly general framework for integrating machine learning and evolutionary search was proposed by Michalski, which we describe below.

2.1 LEM

The Learnable Evolution Model (LEM) [16, 17] integrates a symbolic learning component within evolutionary search; it seeks out rules (or other predictive models) 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 proven able to significantly improve efficiency on a variety of test problems. It is easy to describe the general shape of LEM. Given a machine learning method, and an EA, LEM consists of *Evolution* phases interrupted by *Learning* phases. During an *Evolution* phase, a normal EA is running. For the purposes of LEM, we can regard the result of one of these phases to be a collection of individuals with their associated fitnesses (i.e. all or a subset of those visited by the EA). During a *Learning* phase, this result of the most recent *Evolution* phase is used as training data by a machine learning method, which then learns a model which can discriminate between good and bad solutions. The details are highly general and there is as yet no theory concerning the best way to do this; however, in LEM work so far, a rule-induction method is used to learn a set of rules which are able to predict whether a genome's fitness is above a given threshold (a *positive* rule), and a similar set of rules is learned which capture *bad* solutions, whose fitness is therefore predicted to lie below a given threshold. These rules are then made use of in the generation of an initial population for the next *evolution* phase, and so it repeats. A particular instantiation of a LEM method involves deciding on what machine learning method to use, precisely how to use the rules to help generate the initial population of the next *Evolution* phase, and deciding when to switch between phases.

2.2 Adaptation to Multi-objective Genetic Algorithm

Our framework for incorporating LEM into MO search is described next. We first note that our framework can be used with any MO algorithm. The only requirement is that the MO algorithm maintains an up-to-date Pareto set after each generation.

We have designed LEMMO (Learnable Evolution Model for Multiobjective Optimization), which is designed to seek rules which discriminate between good and bad solutions from the multiobjective viewpoint. In LEM, this discrimination is based purely on thresholds associated with a single-objective fitness; the

```
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
```

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

class of good solutions is simply called *good*, and the class of poor solutions is simply termed *bad*. In LEMMO, there are various possible ways to do this, and our Experiments section investigates some of these. In one method, for example, we only use individuals on the current approximation to the Pareto front as the *good* set, and individuals of rank three or below (in the non-dominated sorting sense) as the *bad* set; section 4 provides details of the different variants studied here.

The rules learned are then used to help create (and/or filter) new individuals, by seeking solutions that match the positive rules and don't match the negative rules. There are several ways to design and apply such a mechanism, and part of our ongoing research is to find the ideal ways of doing this for a range of water system design problems.

LEMMO is designed to use any machine learning method, but particularly attunded to using a rule induction algorithm, in order to produce rules to create new individuals. LEM publications so far have used the AQ learning algorithm. Although existing implementations, such as AQ11 [18] and AQ15 [19] handle noise with pre and post-processing techniques, the basic AQ algorithm heavily depends on specific training examples during search (the algorithm actually employs a beam search). Rather than using a sophisticated version of AQ, we decided in these initial experiments to use a rule induction algorithm with which we are more familiar, and which does not suffer from the lack of robustness to specific training examples as badly as basic AQ; we therefore used the well-known C4.5 decision tree induction algorithm [22], combined with 'C4.5rules' (which extracts rules from the resulting tree), which is provided with the C4.5 code distribution. An example of a rule generated in our application is given figure 2. The features in a rule (A3, A7 etc ...) refer to specific pipe connections in a water network, and the choices for such a feature are potential diameters for the pipe in question (see next section).

After learning a set of rules in the Learning stage of LEMMO, we generate several new matching individuals for each rule that is of the class good (called a *positive rule*). For example, an individual which matches the schema ******9**3*********** will match rule R3. Typically, only a few parts of the genotype are 'fixed' by a rule; the rest of the genotype is (in this study) filled

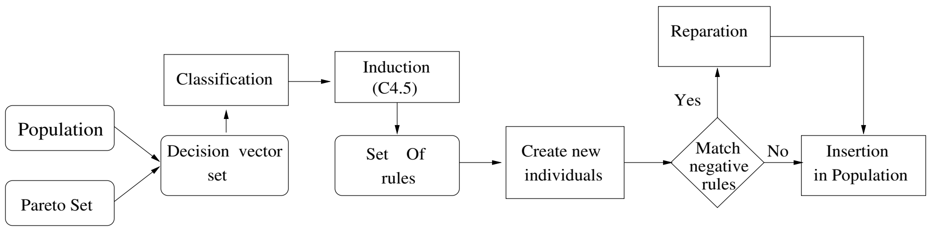

Fig. 3. Creation of new individuals using learned rules

in via one-point crossover of an arbitrary pair of selected parents, ensuring that the parts fixed by the rule are protected. Following this. we verify that the new generated individuals do not match any negative rules, repairing any individual which does. Figure 3 shows the entire process.

3 Application to Water System Design

Water distribution network modeling is used for a variety of purposes. Computer modeling of water distribution networks continues apace in the water industry as computers become increasingly powerful, and more complex systems need to be modeled. Open source software such as EPANET [24] is often used to model water systems, and indeed EPANET is what we used in the current research. In recent years, computational methods such as non-linear programming, dynamic programming and search techniques have been used to optimise the design, operation and rehabilitation of these networks, however MOEA methods, due to their general success in this arena, are becoming increasingly favoured.

Optimisation problems in this area are usually either 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). The test problems used in this article are of the first and second types respectively, and both are regularly used in the water systems research community.

We note that EPANET can be criticised for its suitability and accuracy in certain contexts. However, at this stage we feel that such problems outweighed by the fact that EPANET is freely available, and commonly used as a platform for this field of research.

3.1 NY Problem

The first test problem we use is the the New York Tunnels pipe network (NYT). The objective of the NYT 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,

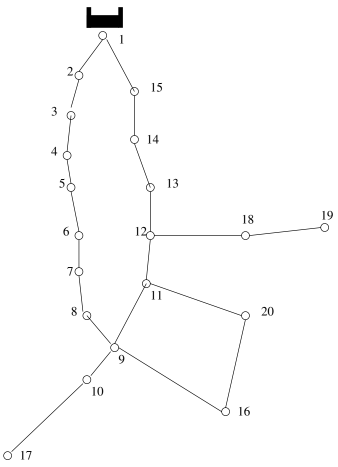

Fig. 4. The New York City problem

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 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 has 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 lenght L_{ij} is in feet. A full enumeration of all possibilities would require: $16^{21} = 1.9342 \times 10^{25}$. From an optimisation perspective, the objective of the NYT 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 modelling software and parameters used.

3.2 Hanoi Problem

Our second, larger test problem concerns a water distribution network in Hanoi, Vietnam, is considered in this study. The network [8] consists of one reservoir (node 1), 31 demand nodes and 34 pipes (see figure 5). The minimum pressure head required at each node is 30 m. The cost of commercially available pipe sizes (12, 16, 20, 24, 30, 40; in inches) was calculated using the equation provide in [8]): $C = 1.1 \times D_{ij}^{1.5} \times L_{ij}$.

4 LEMMO for Water Distribution Systems

Our comparative MOEA is NSGA-II [3]. That is, the comparative algorithm is NSGA-II, while the variants of LEMMO are all NSGA-II hybridised with

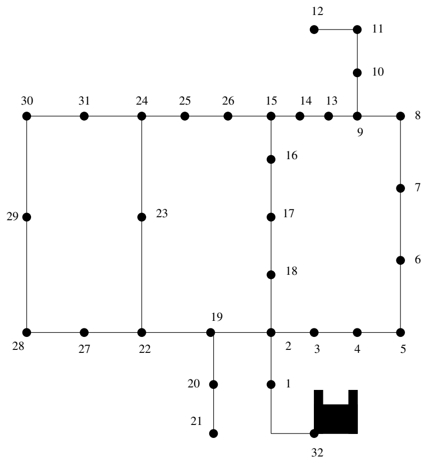

Fig. 5. Hanoi problem

phases in which rules are learned (by the process described above) leading to the generation of an initial population for the next phase of evolution (i.e. NSGA-II).

4.1 Implementation and Parameters

In all cases, the encoding is straightforward, with one individual containing one gene for each pipe, representing a choice of diameter for that pipe. In all cases we use as operator: one-point crossover and single-gene 'random new allele' mutation.

The population size is always 100, the crossover rate is 0.9, the mutation rate is 0.9, and in every trial of every experiment the maximum number of evaluations was set to 250,000.

4.2 LEMMO Variants

For all the explored schemes, the *Learning* is executed on decision vectors (here the size of the pipe) and the value of the objectives is used for the classification task. During the classification phase, each decision vector is put into a class (*bad or good*) according to different strategies that are explained below. This classification is re-done before each call of the induction algorithm in order to avoid conflict between rules.

The different LEMMO variants explored are:

– LEMMO-1: run *Learning* when there has been no change to the approximation to the Pareto front for two successive generations.
– LEMMO-fix1: run *Learning* every 10 generations, using the initial population of the previous *Evolution* phase as the *bad* set and the final population of that phase as the *good* set.

– LEMMO-fix2: run *Learning* every 10 generations, using the 20 individuals most recently inserted into the Pareto set as the *good* set, and the remaining individuals of the current population as the *bad* set.
– LEMMO-fix3: run *Learning* every 10 generations, using randomly chosen individuals from the current approximation to the Pareto set as the *good* set, and the remaining individuals of the current population as the *bad* set.
– LEMMO-fix4: run *learning* every 10 generations, using the best 30% of individuals found so far on one of the objectives (randomly chosen in each learning phase) as the *good* set, and the worst 30% of individuals found so far on that objective as the *bad* set.

4.3 Quality Measures

As suggested in [14], the S metric [30] and the R metrics [13] are generally better than other metrics in terms of their properties with respect to sensitivity to scaling, compatibility with common-sense 'outperformance' relations [13], cycle-inducing properties (i.e. some metrics may rate set A as better than set B, set B as better than set C and set C as better than set A, which is clearly undesirable), and computational overhead. We mainly use the S metric in this study (although it has a rather high computational overhead, this is not such a problem when $k = 2$ and the non-dominated sets are not overly large), but also supplement our results with the use of the R_{1R} metric. Regarding the S metric (details in Zitzler in [30]), its implementation requires a reference point Z_{ref}. In this work we calculate Z_{ref} by using the worst possible solutions in terms of both objectives. For the cost objective, we use the cost arising from using the largest (most expensive) diameter for each pie, and for head deficit objective we take the value resulting from using the minimum diameter for each pipe. When we use the R_{1R} metric (details in [13] and summarised in [14]), this requires a reference non-dominated set, for which we choose an arbitrary result from an NSGA run.

5 Results

In our first set of experiments the aim was to explore a number of parameter variants of our basic LEMMO method, in order to establish a feel for the balance required between learning and evolutionary search on the smaller of the two test problems. Note that even though the NYT and Hanoi water system test problems are relatively small in terms of design variables (compared to other real water system design problems), the computational cost of fitness evaluation remains very great, so we needed to limit the number of experiments. We therefore performed our initial experiments on the smaller of the two problems (NYT), and used a second set of experiments to begin to explore scale-up properties by testing the best variant from the first set of experiments on the larger problem (Hanoi).

5.1 Results on the NYT Problem

We did 30 trials each on the NYT problem with each of the five LEMMO variants and NSGA-II. The obtained approximate Pareto fronts are highly populated. Table 1 gives the S metric statistics, showing the percentage improvement (positive values) of the given LEMMO scheme over the corresponding value achieved by NSGA-II. For example, the median S metric value achieved by LEMMO-fix4 was 9.12% better than the median achieved by NSGA-II. We can see that the LEMMO methods, particularly LEMMO-fix4, led to better final approximate Pareto set quality than NSGA-II. The standard deviation column shows the percentage improvement of the given LEMMO scheme over the standard deviation for NSGA-II. It is notable that in each case the standard deviation is considerably reduced, suggesting that the LEMMO methods not only achieve better quality results, but do so more reliably than NSGA-II. Overall, LEMMO-fix4 seems to be the clear winner. For both LEMMO-fix3 and LEMMO-fix4, the improvement in the median and mean was found to be significant at a 95% confidence level based on a randomisation test [5]. For the $R1_R$ metric (see table 3), we observe that all the schemes are better than NSGA-II.

Table 2 presents statistics on the number of evaluations required to obtained the final Pareto front over the 30 runs for each scheme. That is, the number of evaluations at which further improvements to the Pareto front ceased; this is essentially a convergence measure. Meanwhile, table 2 shows the comparison of this measure between each of the LEMMO schemes and NSGA-II. For example, the median convergence time for LEMMO-fix1 was 16.28% faster than the median convergence time for NSGA-II. Generally, the LEMMO schemes are not always faster than NSGA-II, however this must of course be traded off against the fact that (as indicated above) the occasional slower convergence does lead to better results than NSGA-II. However, LEMMO-fix4 again turns out to be the best in this sense, achieving reliably faster convergence, and to better results. It is worth examining figure 6, which illustrates the evolution of the S metric value over time for LEMMO-fix1 and for NSGA-II. The LEMMO scheme is always ahead of NSGA-II in quality for a given time.

Table 4 presents 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 NGSAII in median (-16.66%) and in mean (-20.58%) to obtain this value.

If we have a look to the solutions generated by LEMMO-fix4, we can notice that the best mono-objective solution obtained in [20, 25] is in all the obtained Pareto sets. The size of the pipes is the same than in [25].

Table 1. Quality assessment S metric (S(Scheme) - S(NSGAII))/S(Scheme)

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%

Table 2. Statistic on number of evaluations for finding the final Pareto front over 30 runs for the different tested schemes (T(Scheme)-T(NSGAII)/T(Scheme))

Scheme	Median	Mean	Min	Max	Std Dev.
LEMMO-1	+14.07%	+6.02%	+29.8%	-1.84%	+4.11%
LEMMO-fix1	**-16.28%**	**-11.81%**	-10.04%	-0.93%	+1.26%
LEMMO-fix2	-8.97%	-4.54%	+20.69%	-0.00%	+5.22%
LEMMO-fix3	-14.69%	-7.38	-14.41%	-0.36%	+8.53%
LEMMO-fix4	-16.14 %	-8.45 %	+25.40%	-0.44%	-21.74%

Table 3. 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

In tables 5 and 6, we show for the different possibilities of the parameters for each scheme the quality of the metrics. We observe that the choice of 10 was a good one.

5.2 Results on the Hanoi Problem

Having found that that the LEMMO schemes, but particularly LEMMO-fix4, seem capable of improving speed (and quality) of results, we did further experiments with LEMMO-fix4 on a larger problem, the Hanoi network [8, 7]. The larger size of this problem (and other time constraints) mean that at the moment we can only report on results of 10 trial runs on this problem for each of NSGA-II and LEMMO-fix4. Table 7 shows the results in terms of the $R1_R$ metric (in this case, lower values are better). It is clear that LEMMO-fix4 achieved far better quality results than NSGA-II on this larger problem, and we find the comparisons between mean and median are significant at a 90% confidence level based on a randomisation test [5].

We also compared LEMMO-fix4 and NSGA-II on this problem in terms of their speed in attaining certain chosen values of the S metric (we chose the S

Table 4. 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

	S_{norm} >**0.955**		S_{norm} > **0.968**	
Scheme	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 5. 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 6. 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

	S_{norm} >**0.955**		S_{norm} > **0.968**	
Scheme (parameter)	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%

Table 7. Quality assessment $R1_R$ metric with a front of NSGA-II for Hanoi problem on 10 experiments

Scheme	$R1_R$ Median	$R1_R$ Mean	$R1_R$ min	$R1_R$ max	$R1_R$ Std Dev.
NSGA II	0.05439	0.06871	0.00199	0.02745	0.07335
LEMMO-fix4	0.01247	0.01854	0.00598	0.03992	0.01261

metric for this for pragmatic reasons concerning our current implementation). We found that to obtain 0.711, LEMMO-fix4 required on average 27.5% fewer evaluations than NSGA II. Meanwhile, LEMMO-fix4 obtains the final S metric 0.7488 in 6 out of 10 experiments, but this is never achieved by NSGA-II within the 250,000 evaluations limit.

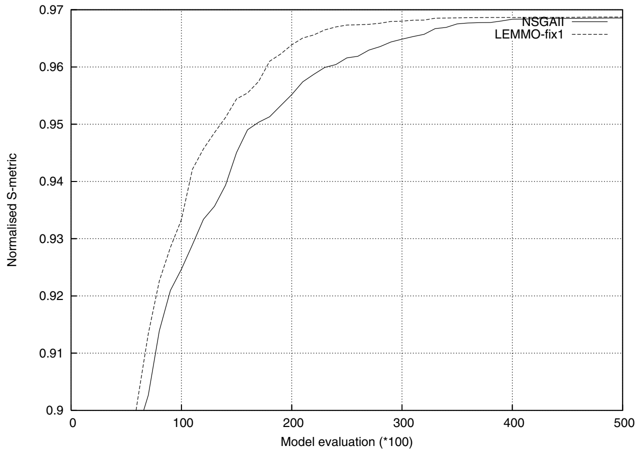

Fig. 6. Example of evolution of the normalized S-metric during the evolution against model evaluation on NY Problem

6 Conclusion

We have reported on initial research into ways to find good solutions to multiobjective water systems design problems in fewer evaluations, with the ultimate goal of enabling multiobjective EAs to be applied regularly to large scale such problems. In particular, inspired by impressive speedup results reported on other (but single-objective) problems we have started to explore whether the LEM method [16, 17] can be successfully used in this domain. Our adaptation of this technique, called LEMMO, and in particular the variant in which the learning method focuses on single-objective values (but alternates between objectives), has indeed been found to significantly improve both the speed and quality of solutions to two benchmark water systems design problems. The fact that LEMMO-fix4 was the better variant suggests that it is very hard to capture, in simple rules, characteristics of solutions which relate to their distance from the Pareto front (which was required of the other LEMMO variants); this indeed seems intuitively fair. We therefore recommend alternating single-objective based learning in such schemes applied to multiobjective problems, although of course the idea of *learning* correlates of distance from the Pareto front remains an intriguing research area.

We were particularly interested in the LEM framework as a way of reducing the required number of evaluations for these problems, however the achievement of better quality results was a welcome added bonus. Indeed it partly makes up for the fact that we would prefer to see more dramatic improvements in speed to achieve a given target. However, we feel that significantly further reduced relative numbers of evaluations will be achieved on larger problems (note that LEMMO-fix4 speedup on the NYT problem was around 16%, and on the larger Hanoi problem the speedup was around 28%), where the 'focussing' effect of the *Learning* phase becomes relatively more imperative. Meanwhile, considering that there is an immense space of possibilities for the way in which the *Learning* phase in LEMMO can be configured (e.g. different learning methods, different ways of generating new solutions from the learned models, and so on), it is very encouraging that these first few attempts have already led to promising results.

Acknowledgment

We are grateful to Joshua Knowles for supplying some of the metrics code, to the EPSRC for the support of Dr Jourdan and to the anonymous referees for their helpful comments.

References

1. P. B. Cheung, L.F.R. Reis, K.T.M. Formiga, F.H. Chaudhry, and W.G.C. Ticona. Multiobjective evolutionary algorithms applied to the rehabilitation of a water distribution system: a comparative study. In C.M. Fonseca, P.J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization (EMO-2003)*, pages 662–676, Springer-Verlag, LNCS 2632, 2003.
2. S.E. Cierniawski, J.W. Eheart, and S. Ranjithan. Using genetic algorithms to solve a multiobjective groundwater monitoring problem. *Water Resources Research*, 31(2):399–409, 1995.
3. 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.
4. J.L. Dorn. and S.R. Ranjithan. Evolutionary multiobjective optimization in watershed quality management. In C.M. Fonseca, P.J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization (EMO-2003)*, pages 692–706, Springer-Verlag, LNCS 2632, 2003.
5. E. Edgington. *Randomization tests.* Marcel Dekker Inc., 1980.
6. M. Erickson, A.S. Mayer, and J. Horn. The niched pareto genetic algorithm 2 applied to the design of groundwater remediation systems. In E. Zitzler, K. Deb, L. Thiele, C. Coello Coello, and D. Corne, editors, *Evolutionary Multi-Criterion Optimization (EMO-2001)*, pages 681–695, Springer-Verlag, LNCS 1993, 2001.
7. K.T.M. Formiga, F.H. Chaudhry, .B. Cheung, and L.F.R. Reis. Optimal design of water distribution system by multiobjective evolutionary methods. In C.M. Fonseca, P.J. Fleming, E. Zitzler, K. Deb, and L. Thiele, editors, *Evolutionary Multi-Criterion Optimization (EMO-2003)*, pages 677–691, Springer-Verlag, LNCS 2632, 2003.
8. O. Fujiwara and D.B. Tung. A two-phase decomposition method for optimal design of looped water distribution networks. *Water Resources Research*, 26(4):539–549, 1990.
9. D. Halhal, G.A. Walters, D. Ouazar, and D. Savic. Multiobjective improvement of water distribution systems using a structured messy genetic algorithm approach. *ACSE Journal of Water Resources Planning and Management*, 123(3):137–146, 1997.

10. 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.
11. H. Handa, T. Horiuchi, O. Katai, T. Kaneko, T. Konishi, and M. Baba. Coevolutionary 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.
12. 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 *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)*, page 764, San Francisco, California, USA, 7-11 July 2001. Morgan Kaufmann.
13. M.P. Hansen and A. Jaszkiewicz. Evaluating the quality of approximations to the non-dominated set. Technical Report TR-IMM-REP-1998-7, Technical University of Denmark, 1998.
14. 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.
15. P. Larranaga and J. Lozano. *Estimation of distribution algorithms: a new tool for evolutionary computation.* Kluwer Academic Publishers, 2001.
16. R.S. Michalski. Learnable evolution model: Evolutionary processes guided by machine learning. *Machine Learning*, 38(1–2):9–40, 2000.
17. R.S. Michalski, G. Cervon, and K.A. Kaufman. Speeding up evolution through learning: Lem. In *Intelligent Information Systems 2000*, pages 243–256, 2000.
18. R.S. Michalski and J.B. Larson. Selection of most representative training examples and incremental generation of vl1 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.
19. R.S. Michalski, I. Mozetic, J. Hong, and N. N. Lavrac. he 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.
20. L.J. Murphy, A.R. Simpson, and G.C Dandy. Pipe network optimisation using an improved genetic algorithm. Technical report, Dept of civil and environment engineering, University of Adelaide, Australia., 1993.
21. M. Pelikan, D.E. Goldberg, and E. Cantu-Paz. Boa: The bayesian optimization algorithm. In W. Banzhaf, J. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M. Jakiela, and R.E. Smith, editors, *Proceedings of the genetic and evolutionary computation - GECCO-99*, pages 525–532, Morgan Kaufmann, 1999.
22. J. Ross Quinlan. *C4.5: programs for machine learning.* Morgan Kaufmann Publishers Inc., 1993.
23. P.M. Reed, B.S. Minsker, and D.E. Goldberg. A multiobjective approach to cost-effective long term groundwater monitoring using an elitist nondominated sorting genetic algorithm with historical data. *Journal of Hydroinformatics*, 3(2):71–89, 2001.
24. L.A. Rossman. Epanet, users manual. Technical report, U.S. Envi. Protection Agency, Cincinnati, Ohio, 1993.
25. D. Savic and G. Walters. Genetic algorithms for least-cost design of water distribution networks. *Journal of Water Resources Planning and Management*, 123(2):67–77, 1997.

26. D. Savic, G.A. Walters, and M. Schwab. Multiobjective genetic algorithms for pump scheduling in water supply. In D. Corne and J. Shapiro, editors, *AISB Workshop on evolutionary computation - selected papers*, pages 227–236, Springer-Verlag, LNCS 1305, 1997.
27. M. Sebag, C. Ravise, and M. Schoenauer. Controlling evolution by means of machine learning. In *Evolutionary Programming*, pages 57–66, 1996.
28. 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.
29. 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.
30. E. Zitzler. Evolutionary algorithms for multiobjective optimization: Methods and applications. Master’s thesis, Swiss federal Institute of technology (ETH), Zurich, Switzerland, november 1999.