Performance of LEMMO with artificial neural networks for water systems optimisation

William Sayers[a*], Dragan Savic[b,c] and Zoran Kapelan[b,d]

[a]School of Business and Technology, University of Gloucestershire, Cheltenham, UK;
[b]Centre for Water Systems, University of Exeter, Exeter, UK

[c]KWR Water Cycle Research Institute, Nieuwegein, the Netherlands

[d]Delft university of Technology, Faculty of Civil Engineering and Geosciences, Department of Water Management, Delft, The Netherlands.

William Sayers, wsayers@glos.ac.uk; Dragan Savic, d.savic@exeter.ac.uk, dragan.savic@kwrwater.nl; Zoran Kapelan, z.kapelan@exeter.ac.uk, z.kapelan@tudelft.nl;

Artificial neural networks and LEMMO for water systems optimisation

Optimisation algorithms could potentially provide extremely valuable guidance towards improved intervention strategies and/or designs for water systems. The application of these algorithms in this domain has historically been hindered by the extreme computational cost of performing hydraulic modelling of water systems. This is because running an optimisation algorithm generally involves running a very large number of simulations of the system being optimised. In this paper, a novel optimisation approach is described, based upon the "learning evolution model for multi-objective optimisation" algorithm. This approach uses deep learning artificial neural network meta-models to reduce the number of simulations of the water system required, without reducing the accuracy of the optimisation results. This is then compared to an industry standard optimisation approach, showing results with increased speed of convergence and equivalent or improved accuracy. Therefore, demonstrating that this approach is suitable for use in highly computationally demanding areas such as water systems optimisation.

Keywords: optimization; flooding; water-distribution

Introduction

The application of optimisation algorithms to water systems has been hindered by the extreme computational cost of hydraulic system modelling, although a large amount of research in this area has been undertaken, increasing in quantity in the last 5-10 years (Bach et al., 2014; Djordjević et al., 1999; Garcìa et al., 2015; Hammond et al., 2018; Sayers, 2015; Sayers et al., 2014; Shirzad, 2017; Shishegar et al., 2018; Wang et al., 2014; Webber et al., 2018; Woodward, 2012; Woodward et al., 2013a, 2013b; Zheng et al., 2019). The run time of hydraulic system models can vary considerably, depending on the exact model and the hardware running it. However, runtimes in the order of sixty seconds per model run, for a reasonably complex model, would not be unreasonable. Assuming this runtime, if one hundred thousand evaluations were required (as is the case for the simplest networks we have tested with) the total runtime would be approximately 70 days. In reality, due to model complexity this is likely to be significantly under-estimating the time necessary. It is because of this computational limitation that research such as that by Shirzad (2017) on shortening search times in water distribution optimisation, or Webber et al. (2018) on more rapid assessment of flood management options are of such importance to the field.Various optimisation algorithms have been tested for suitability, accuracy of results, and speed of convergence, amongst other performance measures (Behzadian et al., 2009; Shirzad, 2017; Shishegar et al., 2018; Wang et al., 2014; Wang, 1997; Webber et al., 2018; Woodward, 2012; Woodward et al., 2013a, 2013b; Zheng et al., 2017).

In this paper we present an optimisation algorithm based on the learning evolution model for multiple-objective optimisation (LEMMO) (Jourdan et al., 2004, 2005) combined with feed-forward artificial neural network (ANN) meta-models (Behzadian et al., 2009; Mohtar et al., 2018; Sayers, 2015; Sayers et al., 2014). Along with this a rigorous examination of the effectiveness of this algorithm when applied to selected water distribution systems test-cases described in Wang et al. (2014) and compared to reference pareto fronts generated in the same paper is also presented.

Methodology

LEMMO-ANN Overview

The basis of the multi-objective optimisation in the described algorithm is the industry-standard and well-used NSGA-II algorithm (Bekele and Nicklow, 2007; Deb et al., 2000, 2002; Kannan et al., 2009). In order to reduce the computational impact of objective function evaluation, and therefore increase the speed and accuracy with which a reasonable Pareto front can be achieved, NSGA-II can be combined with heuristic meta-models (Behzadian et al., 2009; Sayers et al., 2014; Sayers, 2015).

The technique used in this research paper is based upon the LEMMO algorithm (di Pierro et al., 2009; Jourdan et al., 2004, 2005), specifically a modification of "LEMMO-fix4", which is in turn based upon the LEM algorithm (Michalski et al., 2000; Wojtusiak and Michalski, 2006). LEMMO-fix4 is a multi-objective version of LEM, which utilises decision trees to identify rules that characterise promising solutions. The LEMMO algorithm works by performing a number of iterations of standard NSGA-II (Deb et al., 2000, 2002) and storing data on which of the generated solutions are "good", and which are "poor" in respect to a randomly chosen objective from the objectives specified. This stored data is then used to train a decision tree algorithm to distinguish between "good" and "poor" solutions, using the best and worst 30% as "good" and "poor" sets. New solutions are then generated which the decision tree categorises as "good". These generated solutions are then integrated with the main population of the optimisation algorithm (allowing them to be culled if existing solutions are better) and the algorithm continues.

The implementation described in this paper uses a similar approach, but with artificial neural networks taking the place of decision trees within the algorithm (Sayers, 2015) in an attempt to improve the performance of the original algorithm when applied to computationally intensive problems such as water systems optimisation. This is described in detail below.

This approach is designed to give the incremental improvement behaviour of the original optimisation algorithm, but also allows the algorithm to make intuitive bursts when it discovers promising new strategies (Jourdan et al., 2004, 2005; Sayers, 2015). This is by virtue of the machine learning techniques recognising new high-scoring solutions and "seeding" the population with variants of these.

Optimisation objectives for LEMMO-ANN testing

In common with other optimisation algorithms, the formulation of the objective function is of paramount importance, the problem must be represented in a way that captures all essential elements, is differentiable, and is easily modifiable.

In the case of the algorithm described in this paper, LEMMO with artificial neural networks (LEMMO-ANN) two objective functions minimum are required for a meaningful test of the algorithm.

In (Wang et al., 2014) a number of reference Pareto fronts are generated using benchmark water distribution system problems. In order to compare with these reference Pareto fronts, when applied to water distribution system problems, we are using network resilience (Prasad and Park, 2004; Wang et al., 2014) and capital expenditure as our objectives. The formulation of these objectives can be seen in the following equations:

$$\min C = \sum_{i=1}^{np} a \times D_i^b \times L_i \qquad (1)$$

$$\max I_n = \frac{\sum_{j=1}^{nn} C_j Q_j \left(H_j - H_j^{req}\right)}{\left(\sum_{k=1}^{nr} Q_k H_k + \sum_{i=1}^{npu} \frac{P_i}{\gamma}\right) - \sum_{j=1}^{nn} Q_j H_j^{req}} \qquad (2)$$

$$C_j = \frac{\sum_{i=1}^{npj} D_i}{npj \times \max\{D_i\}} \qquad (3)$$

In equation 1 ‘C’ represents total cost (monetary units are problem dependant); ‘np’ represents the total number of pipes; ‘a’ and ‘b’ represent constants depending on specific problems; ‘D_i’ is the diameter of pipe ‘i’ and ‘L_i’ is the length of pipe ‘i’ (Wang et al., 2014).

In equations 2 and 3, ‘I_n’ is network resilience; ‘nn’ represents number of demand nodes; ‘C_j’, ‘Q_j’, ‘H_j’ and ‘H_j^{req}’ represent uniformity, demand, actual head, and minimum required head of node ‘j’; ‘nr’ is number of reservoirs; ‘Q_k’ and ‘H_k’ are discharge and actual head of reservoir ‘k’; ‘npu’ is the number of pumps; ‘P_i’ is the power of pump ‘i’; ‘γ’ represents the specific weight of water; ‘npj’ is the number of pipes that are connected to node ‘j’; ‘D_i’ is the diameter of pipe ‘i’ connected to demand node ‘j’ (Wang et al., 2014).

The LEMMO-ANN algorithm

The optimisation algorithm implemented is based on the NSGA-II algorithm (Deb et al., 2000, 2002) and the LEM (Wojtusiak and Michalski, 2006) and LEMMO algorithms (di Pierro et al., 2009; Jourdan et al., 2004, 2005), specifically LEMMO-fix4. Rather than the decision-tree system used as a machine learning meta-model in the original LEMMO implementation, a feed-forward artificial neural network is employed.

The algorithm performs a number of iterations of standard NSGA-II (Deb et al., 2002), storing data on which of the generated solutions are “good” solutions and which are “poor”. For this purpose, “good” solutions are those within the top 30% when ordered by rank and then crowding distance, “poor” are those within the bottom 30% with the same ordering. It then uses this data to train a machine-learning algorithm as a classifier to distinguish between good and bad solutions and uses the resulting network from this training to generate new solutions which are classified as “good” via a search process. These are then integrated with the main population in the same manner as newly generated solutions are in the standard algorithm. Execution then continues with further iterations based on pure NSGA-II and further iterations of machine learning as described in LEMMo-fix4 (Jourdan et al., 2005).

This approach is driven by the knowledge that for certain classifications of problem, ANNs have several potential advantages over decision trees. Firstly, ANNs are universal approximators (Cybenko, 1989; Hartman et al., 1990; Hornik et al., 1989; Hornik, 1991) which cope well with on-line training in a continuous manner. Additionally, ANNs classify data into sets based on non-linear boundaries (Masters, 1993), whereas decision trees are limited to linear boundaries (Quinlan, 1993), which depending on maximum tree-depth, could affect classification accuracy.

Artificial neural network implementation

The artificial neural network used in this project is and is a feed-forward neural network trained by means of a resilient propagation algorithm (RPROP) (Igel and Hüsken, 2000; Riedmiller and Braun, 1993). The implementation in this work is a part of the Accord library (Souza, 2015). A feed-forward neural network was selected as this type of network is robustly proven to be a universal approximator (Cybenko, 1989; Hartman et al., 1990; Hornik et al., 1989; Hornik, 1991; Park and Sandberg, 1991). Additionally, the function of the network, where a given number of inputs are associated with a specified output and the weights are altered to give an input/output mapping for the problem fits well into the context of being used within another algorithm. Multilayer feed forward artificial neural networks degrade in performance gracefully, as the amount of noise in the input increases (Svozil et al., 1997). ANN’s also cope well with being trained online, which is important for the applications detailed in this paper.

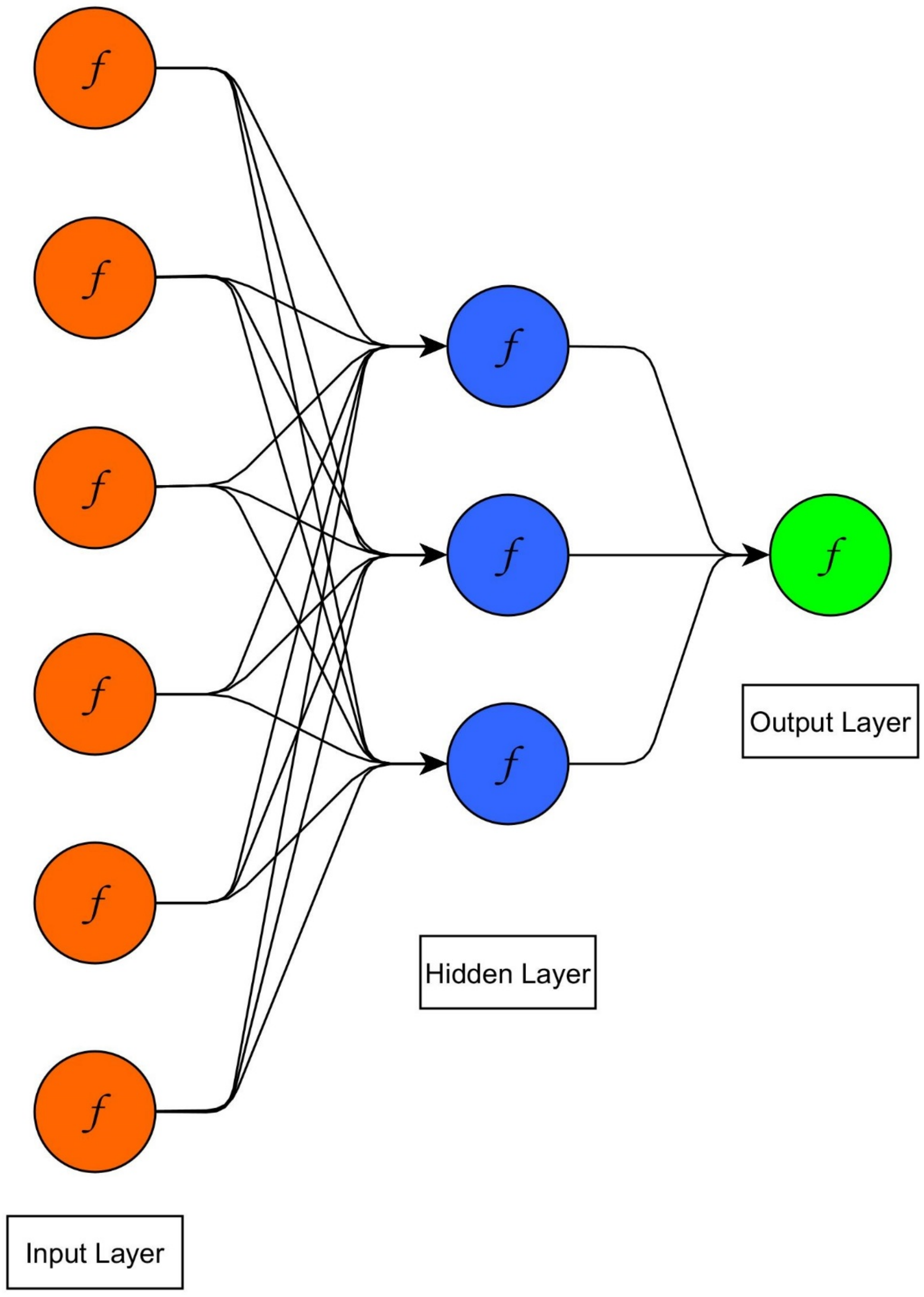
RPROP was chosen as it is a fast and effective alternative to standard back propagation. Due to the way in which the various approaches integrate with the ANN, a large or complex training algorithm is likely to have a significant impact on performance. So something relatively simple, but proven and effective (Igel and Hüsken, 2000; Riedmiller and Braun, 1993) was selected.

Artificial neural network structure

It has been shown that a feed-forward neural network with one hidden layer is a universal approximator (Cybenko, 1989; Hornik et al., 1989; Hornik, 1991) given the correct parameters (i.e. weight values). However, how to find these parameters and how many nodes should be present in a hidden layer is not identified.

Additionally these papers (Cybenko, 1989; Hornik et al., 1989; Hornik, 1991) do not show that the three-layer approach is the most efficient and effective for a given problem, only that it theoretically should be able to achieve an approximation. There is no universal rule or system for selection of the correct number of hidden neurons for a given problem, but most suggested guidelines are between zero and 'n' with 'n' being the number of decision variables. Initially, therefore, a three-layer network was used where the number of hidden nodes was equal to the number of decision variables divided by two (see Figure 1).

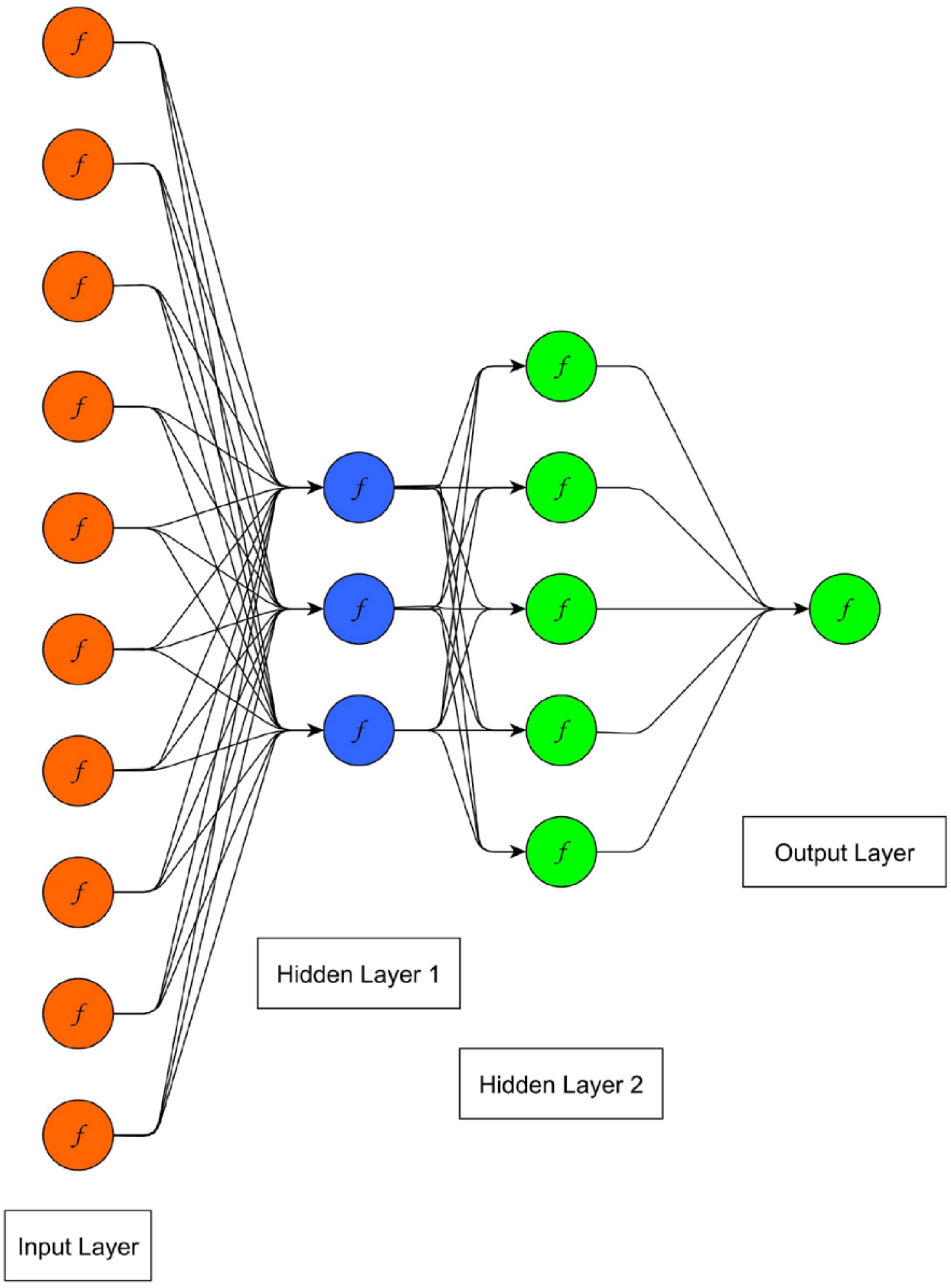
Figure 1. Initial neural network structure with six input nodes



The training algorithm used was resilient back-propagation (RPROP) (Riedmiller and Braun, 1993), as previously mentioned. This network structure seemed to produce an improvement in approximating the Pareto front on smaller problems but failed to achieve the same for larger problems. Additionally, by analysing the execution of the code, it could be seen that the neural network-training algorithm was running until it hit a hard limit imposed to prevent endless loops (Sayers, 2015). This was interpreted as meaning that the neural network was struggling to classify the inputs, and was not producing a meaningful enough answer for the LEMMO algorithm to function correctly (Sayers, 2015).

It has been suggested (Chester, 1990; Masters, 1993) that a network with two hidden layers could perform better than one with a single hidden layer when approximating complex discontinuous functions. Based on this, it was decided to experiment with adding an extra layer, comprised of half the number of hidden nodes in the previously existing hidden layer (number of decision variables divided by two) (Chester, 1990; Sayers, 2015). This layer is between the previously existing hidden layer and the input layer (see Figure 2). This network structure produced considerably improved results in terms of convergence, diversity and dominated hypervolume (Sayers, 2015), over the previous arrangement and the results for LEMMO-ANN are based on this network structure.

Figure 2. Graphical representation of the neural network structure with ten input nodes (for illustration only - test-problems and real problems should have more inputs).



Integration of ANN meta-models into LEMMO

The machine learning algorithms used in previous implementations of LEMMO are decision tree classifiers. In this case the machine-learning algorithm is an artificial neural network, which has necessitated some modification of the algorithm. This machine-learning algorithm is then utilised to generate a new population for the next evolution phase to use as a starting point. This functionality had to be adapted somewhat to make it applicable to multiple-objective optimisation.

Five variants were examined in Jourdan's (2005) paper, referred to as LEMMO-1, LEMMO-fix1, LEMMO-fix2, LEMMO-fix3 and LEMMO-fix4.

An approach based upon LEMMO-Fix4 has been used as this was recommended in Jourdan et al. (2005) and testing in di Pierro et al. (2009) led to that study also following this recommendation.

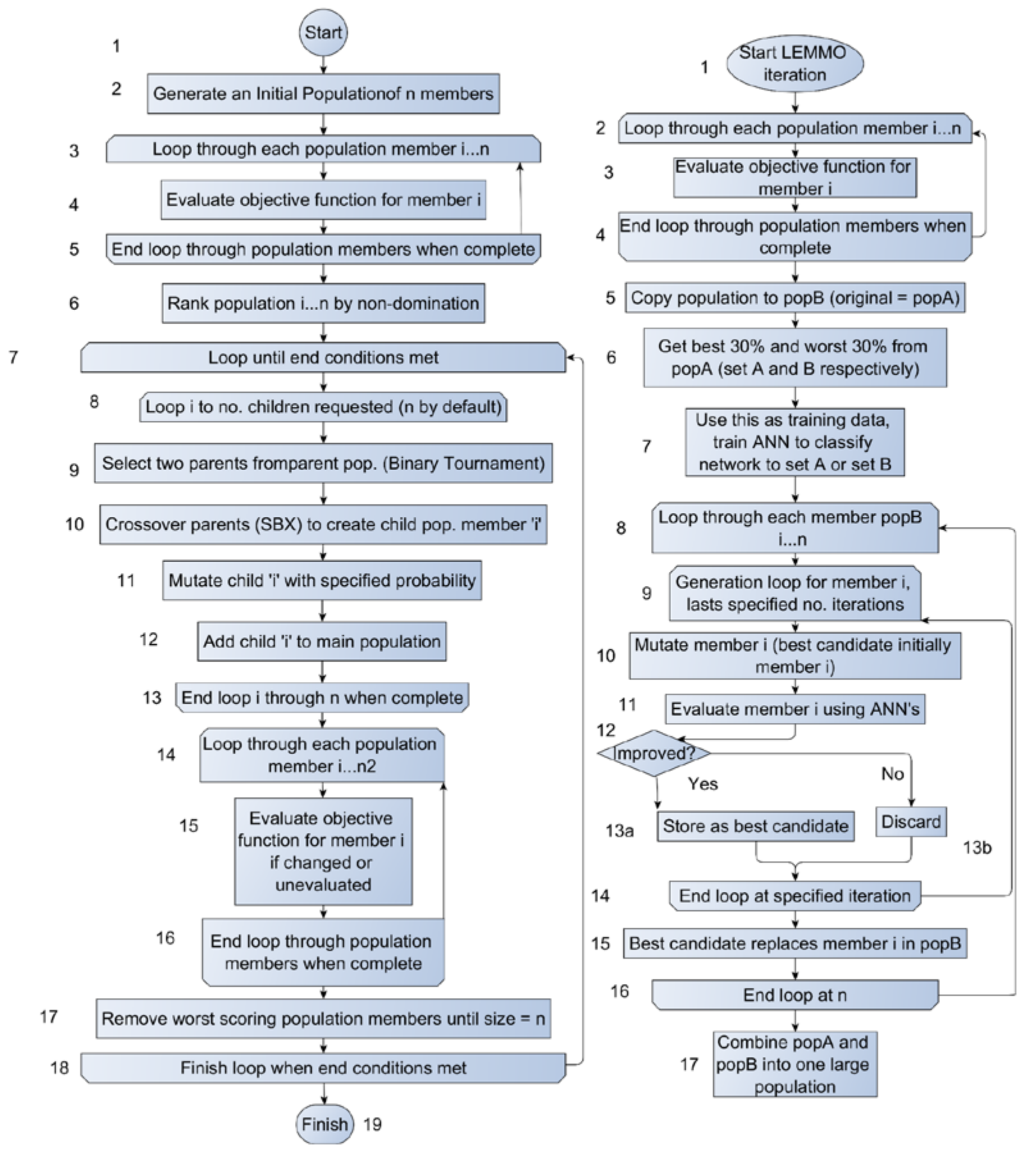
In our approach, the LEMMO-ANN algorithm is run as part of the NSGA2 algorithm, every ten generations.

The feed-forward artificial neural network is trained using RPROP with the best thirty percent of the solutions from the last ten generations as the “good” set and the worst thirty percent as the “poor” set. In order to generate solutions that match the “good” set and do not match the “poor”, solutions are generated for each of the population members in turn.

These solutions are, in a loop, mutated and evaluated (by the ANN), discarding poor mutations and retaining the mutations that improve the solution. At this stage, no solution can enter the population if a solution already exists with the same characteristics. Once a specified number of iterations are completed, the best solution generated so far is retained in that position in the population, and we move onto generating the next.

Finally, this newly generated population is treated as a new child population within the NSGA2 algorithm. This means that a conglomerated population of the current solutions, plus these newly generated solutions is created, evaluated, ranked, analysed for crowding distance, sorted by rank then crowding distance, and the best 50% retained for the next iteration of NSGA2. In this way it is ensured that only improved solutions generated by LEMMO will persist into the optimisation process, the rest will be discarded (see 3).

Figure 3. Implementation of LEMMO-ANN and NSGA2 Algorithm (simplified diagram)



1 Start
2 Generate an Initial Populationof n members
3 Loop through each population member i...n
4 Evaluate objective function for member i
5 End loop through population members when complete
6 Rank population i...n by non-domination
7 Loop until end conditions met
8 Loop i to no. children requested (n by default)
9 Select two parents fromparent pop. (Binary Tournament)
10 Crossover parents (SBX) to create child pop. member 'i'
11 Mutate child 'i' with specified probability
12 Add child 'i' to main population
13 End loop i through n when complete
14 Loop through each population member i...n2
15 Evaluate objective function for member i if changed or unevaluated
16 End loop through population members when complete
17 Remove worst scoring population members until size = n
18 Finish loop when end conditions met
Finish 19
1 Start LEMMO iteration
2 Loop through each population member i...n
3 Evaluate objective function for member i
4 End loop through population members when complete
5 Copy population to popB (original = popA)
6 Get best 30% and worst 30% from popA (set A and B respectively)
7 Use this as training data, train ANN to classify network to set A or set B
8 Loop through each member popB i...n
9 Generation loop for member i, lasts specified no. iterations
10 Mutate member i (best candidate initially member i)
11 Evaluate member i using ANN's
12 Improved?
Yes
No
13a Store as best candidate
Discard 13b
14 End loop at specified iteration
15 Best candidate replaces member i in popB
16 End loop at n
17 Combine popA and popB into one large population

Performance metrics utilised

Three performance metrics will be utilised in evaluation the performance of the algorithm presented in this document, when applied to water distribution systems. These are a diversity metric, a convergence metric, and a measure of dominated hypervolume.

Diversity Metric

The first of the three selected performance metrics is the diversity measure described by Deb et al. (2002). This measure involves calculating the Euclidean distance between each member of the generated Pareto front and its neighbour. The extreme solutions are then calculated in Deb's implementation by fitting a curve parallel to that of the true Pareto-optimal front. The extreme solutions are found by calculating the values of both objectives for the problem in question for two cases. The first case being where all pipes and storage nodes are the maximum allowed size, and the second case being where all pipes and storage nodes are their initial size (i.e. cost will be 0, EAD will be at its starting value).

In equation 4 the process for calculating the diversity metric is described, where "d_f" and "d_l" are the Euclidean distances between the extreme solutions and the boundary solutions of the non-dominated set. Meanwhile "d" represents the average of all distances for the non-dominated set.

$$\Delta = \frac{d_f + d_l + \sum_{i=1}^{N-1} \left| d_i - \bar{d} \right|}{d_f + d_l + (N-1)\bar{d}} \qquad (4)$$

With this measure, lower numbers are better, as they indicate a more uniform spread of solutions along the estimated Pareto front, covering larger areas of the estimated Pareto front. This measure has a benefit in that it can be applied to problems where the true Pareto front is unknown provided one can calculate the extreme end-points of the true Pareto front (Deb et al., 2002).

Convergence Metric

This metric involves measuring how close the various points in a non-dominated set are to another set of coordinates (representing either a true Pareto front, or another estimated Pareto front which is believed to be a superior approximation). It is based upon the measure described in Deb's (2002) paper on NSGA-II. In Deb's original metric a set of 500 uniformly spaced solutions is selected from the superior front. For each calculated solution to be compared, the minimum Euclidean distance of that point from the chosen solutions in the superior front is then computed. The average of all these distances is used as the metric. Therefore, the lower the average of these distances, the better the score.

The issue encountered with Deb's metric is that in a situation where there are fewer solutions on the estimated Pareto front than the true Pareto front a very low value can be obtained. This could give a false impression as to how close to matching the Pareto front an estimated front may be.

A modification has, therefore, been made by the author to overcome this problem. The solutions on the best-known front are taken and for each of those solutions the minimum Euclidean distance to a member of the set of algorithmically generated solutions is identified. The average of those distances is then taken.

The difference can be seen in Table 1 and Figure 4. These tables and figure contain unitless example data, simply to demonstrate the mathematics (units in a real-world application would depend upon the parameters being measured). Table 1 contains the coordinates for data section 'A', as well as the minimum Euclidean distance from each of these points to the points in the Pareto front. The average of these points is 1.21.

Table 1 also contains the coordinates for data section ‘B’ from Figure 4 and, again, the minimum Euclidean distances for these points to the points in the Pareto front. These distances average to 0.35.

The data in these two sections, combined with a visual check on Figure 4, indicates that dataset ‘B’ is a poorer fit than dataset ‘A”. However, because of the different numbers of data points in each dataset, dataset ‘B’ achieves a better convergence value than dataset ‘A’.

On the other hand, in the final sectio the figures for the minimum distances from each data point in the Pareto front, to the data points in ‘A’ and ‘B’, can be seen. These figures average to 1.04 and 2.47 respectively, giving a better estimation of how far from matching the true Pareto front these two datasets are. Much like the original measure, if there is a perfect match (including identical data-points being found) this measure will produce zero. Therefore, the lower the number, the closer the estimated front is to the true Pareto front.

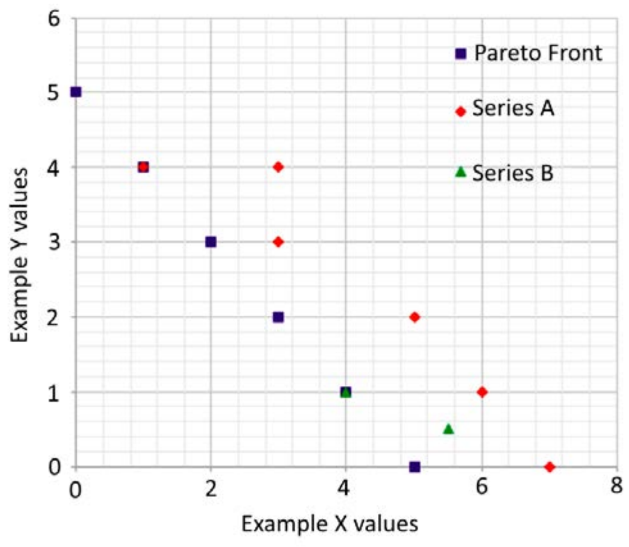
The mathematical expression for this metric can be seen in equation 5 where ‘x’ and ‘y’ are the coordinates for the Pareto front and accented ‘x’ and ‘y’ are the coordinates for the estimated Pareto front.

Table 1. Convergence metric example data 'A'

Example Data 'A'			**Example Data 'B'**		
X	**Y**	**Distances**	**X**	**Y**	**Distances**
7	0	2.00	4.00	1.00	0.00
6	1	1.41			
5	2	1.41			
3	3	1.00	5.50	0.50	0.71
3	4	1.41			
1	4	0.00			
Average Distance:		1.21	Average Distance:		0.35

Example Pareto Front Data			
X	**Y**	**'A' Distances**	**'B' Distances**
5	0	1.41	0.71
4	1	1.41	0.00
3	2	1.00	1.41
2	3	1.00	2.83
1	4	0.00	4.24
0	5	1.41	5.66

Figure 4. Convergence metric example data



$$\frac{\sum_{i=1}^{n}\left(\sum_{j=1}^{m} min\sqrt{\left(x_i - \acute{x}_j\right)^2 + \left(y_i - \acute{y}_j\right)^2}\right)}{n} \qquad (\quad 5 \quad)$$

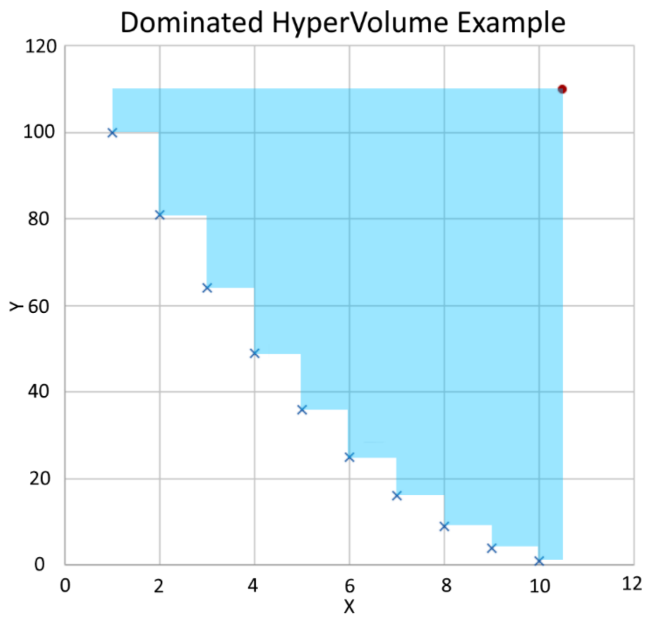
Dominated Hypervolume Metric

The dominated hypervolume or “S-Metric” is a measure of the Hypervolume dominated by the estimated front (Zitzler and Thiele, 1998). Given a reference point, the volume of space dominated by the estimated Pareto front can be calculated, resulting in a measure by which to compare different estimated fronts. The larger the volume of dominated space (i.e. the higher the numerical value of the metric) the better the estimation of the Pareto front. This can be seen in figure 5, where the dominated region is indicated by shading.

The reference has to be in such a position that it will encompass the entire Pareto front to be measured. Additionally, the reference point must be the same between separate tests, if they are intended to be compared. Differing reference points could result in wildly different results. Finally, in the given example (see figure 5) both objectives are being minimized – whereas in our test networks, one objective (resiliency) is being maximised rather than minimised.

We are using the DEAP library implementation of the Hypervolume metric, written in Python (Fortin et al., 2012; Wessing, 2010), executed in the .NET environment alongside C# code using the IronPython python implementation (Foord and Muirhead, 2009). This implementation assumes minimisation on both objectives. This problem was solved by inverting the scores from the particular objective that is being maximised, before applying the hypervolume metric. This results in a “flip” of the curve for that particular objective, meaning a reference point based upon this new, but equivalent, curve can be provided and the algorithm works without issue or alteration.

Figure 5. Dominated Hypervolume example, shaded area represents dominated volume



Discussion and results

Water distribution system test-cases

In the original paper describing these benchmark problems (Wang et al., 2014) twelve water distribution system design problems were examined. They fit into four categories: small, medium, large and very large network problems. This categorisation is based on the size of the search space that is defined by each of these problems (see Table 2).

Table 2. Test problem categories (the size of the search space given in the brackets)

Small	Medium	Large	Very Large
Two-reservoir network (TRN) (3.28×10^{7})	New York tunnel network (NYT) (1.93×10^{25})	Fossolo network (FOS) (7.25×10^{77})	Modena network (MOD) (1.32×10^{353})
Two-loop network (TLN) (1.48×10^{9})	Blacksburg network (BLA) (2.30×10^{26})	Pescara network (PES) (1.91×10^{110})	Balerma irrigation network (BIN) (1.00×10^{455})
BakRyan network (BAK) (2.36×10^{9})	Hanoi network (HAN) (2.87×10^{26})		Exeter network (EXN) (2.95×10^{590})
	GoYang network (GOY) (1.24×10^{27})		

It was considered that at least one small problem should be included to allow for easy bug testing and modification of software. Additionally, a smaller problem has the advantage that the best estimated Pareto front has been found by exhaustive search and is therefore known. A medium problem was then included, in order to ensure that problems were tested across a reasonable range of the complexities available (see Table 2). Finally, two ‘very large’ problems were included, as these most accurately represent the scale and type of problem that the new approach is designed to solve.

Taking these considerations into account the selected problems are the two-loop network (TLN), the GoYang network (GOY), the Modena network (MOD) and the Balerma irrigation network (BIN). The details of these problems can be seen in Table 3.

Table 3. Test problem details

Problem	Water Sources	Decision Variables (Pipes)	Pipe Diameter Options	Search Space Size
TLN	1	8	14	1.48×10^{9}
GOY	1	30	8	1.24×10^{27}
MOD	4	317	13	1.32×10^{353}
BIN	4	454	10	1.00×10^{455}

In the benchmark test paper (Wang et al., 2014) a computational budget was fixed, in order that the results were repeatable easily by maintaining a similar computational budget. The computational budgets used in this paper for the chosen tests can be seen in Table 4. They are represented by a cap on the number of model evaluations allowed. Additionally, tests were performed with varying populations (groups 1, 2, and 3). This is because a smaller population with a hard cap on evaluations will cause a deeper search, and a larger population a broader search.

Table 4. Test problem computational budget in original benchmarking

Problem	Number of Evaluations	Group 1 Pop.	Group 2 Pop.	Group 3 Pop.
TLN	100,000	40	80	160
GOY	600,000	60	120	240
MOD	2,000,000	200	400	800
BIN	2,000,000	200	400	800

The best-known Pareto set was identified in Wang et al. (2014) by running a large number of different optimisation algorithms, conglomerating the results, and identifying the best non-dominated set from those conglomerated results. Because of this all the results within the best-known Pareto front were not generated using NSGA-II, and it was not expected that during our testing the algorithm would identify every single result that the Wang et al. (2014) identified. The number and percentage (against the overall total) of solutions identified by NSGA-II in the best known-Pareto fronts for each problem selected can be seen in Table 5 and Table 6, respectively.

Table 5. Contribution to best-known Pareto front from NSGA-II (Wang et al., 2014)

Problem	Group 1 Contribution	Group 2 Contribution	Group 3 Contribution
TLN	54	74	77
GOY	4	23	31
MOD	71	61	26
BIN	8	67	179

Table 6. Percentage contribution to the best-known Pareto front from NSGA-II in percentages (Wang et al., 2014)

Problem	Total Solutions in Best-Known Pareto front	Percent Discovered by NSGA-II (%)
TLN	77	100
GOY	67	43.3
MOD	196	57.7
BIN	265	72.5

In Table 5 the number of contributions to the best-known Pareto front can be seen from each NSGA-II group run within Wang's tests. It can be seen in these results that certain problems seem to lend themselves to higher populations, which means a broader exploration of the available search space. Meanwhile, other problems lend themselves to smaller populations but necessarily higher numbers of iterations (to keep to the same computational budget), which means a deeper exploration of the available search space. With regard to the very large problems one of each of these variants is included (the MOD and BIN problems).

Additionally in Table 6 it can be seen that during Wang et al. (2014)'s tests, NSGA-II performs very well on TLN, more poorly on MOD and GOY, and then better again on BIN. This pattern is mirrored by the NSGA-II implementation we have developed, as would be expected.

Test-case approach

In order for the two different algorithm types to be analysed for each of the four problems, twenty tests were run for each, first without and then with LEMMO-ANN.

For each of these twenty tests, two tests that are a reasonable representation of the overall results were selected to be shown in more detail in this chapter. Each of these selected tests were separated into evenly spaced iterations, in order to show clear progression of the optimisation in the way which would be expected. A visual comparison of these tests also holds some value. It may be worth noting that for every single iteration, of every single result, a graph was generated and inspected. However, for brevity's sake, not all were included.

Test-case analyses

Three metrics have been utilised in the analysis of these results, these are convergence, diversity, and dominated hypervolume. For each test WDS problem there are twenty runs of the tested algorithm, for which the results for every tenth iteration of that run have been recorded.

All results from all twenty runs are included within this analysis. This is achieved by calculating all three metrics for each tenth iteration of every single test. These metrics for each iteration are then averaged across common tests. For example, the metric results for TLN with NSGA-II and no LEMMO consist of a set of averages of the metrics produced for each iteration.

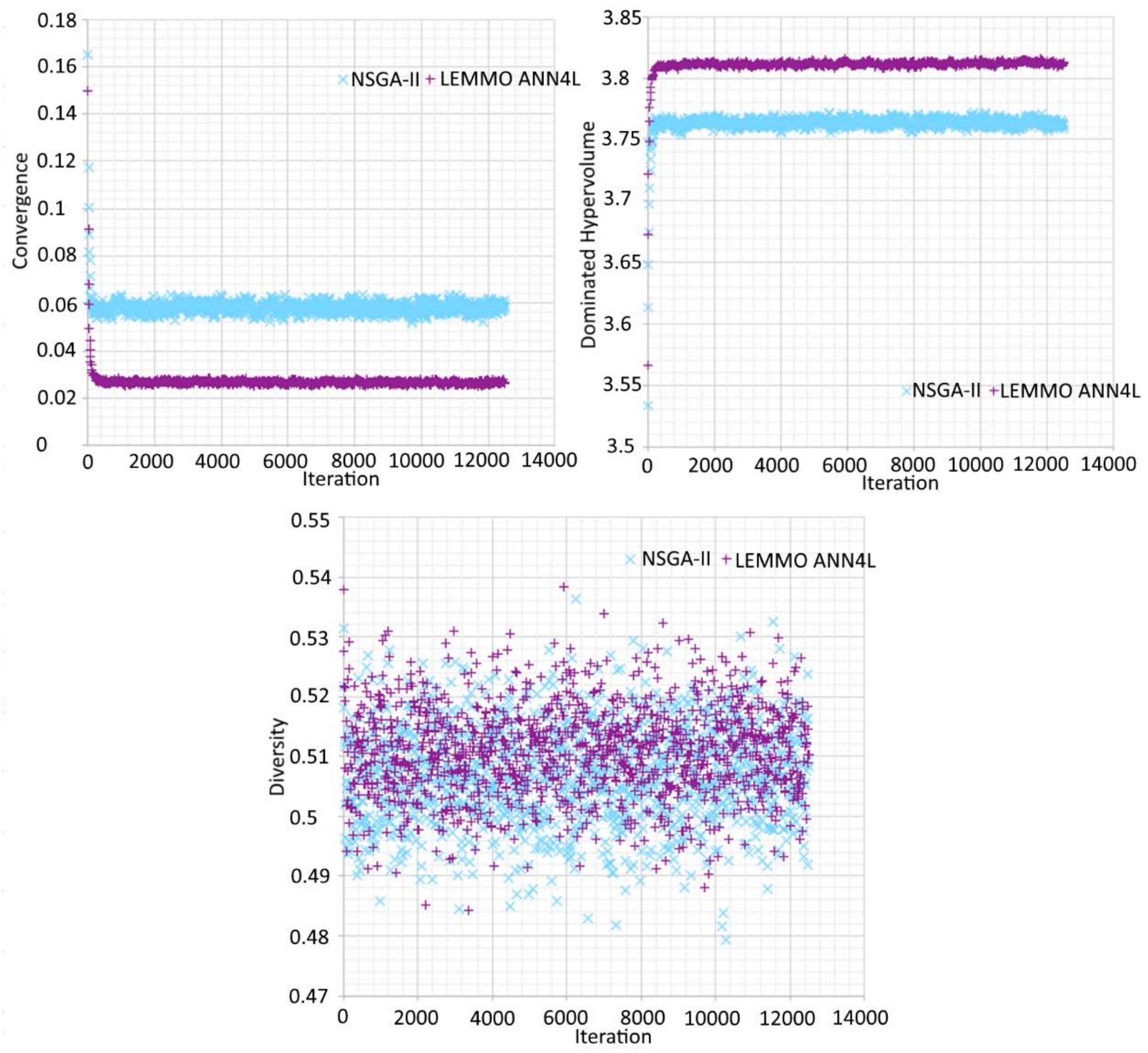
TLN

The analysis for TLN (see figure 6) shows that in terms of convergence towards the best known Pareto front, LEMMO tests show distinctly improved results over the NSGA-II base algorithm.

In terms of diversity, both LEMMO-ANN and the NSGA-II base algorithm start off at a diversity of approximately point five. This diversity then decreases slightly but remains fairly static for the duration of the run algorithm. The NSGA-II base algorithm tends towards very slightly lower diversity throughout, probably explained by the lack of meta-model generated solutions that would be present in the LEMMO-ANN algorithm.

In terms of dominated hypervolume, LEMMO-ANN out-performs the NSGA-II base algorithm fairly decisively.

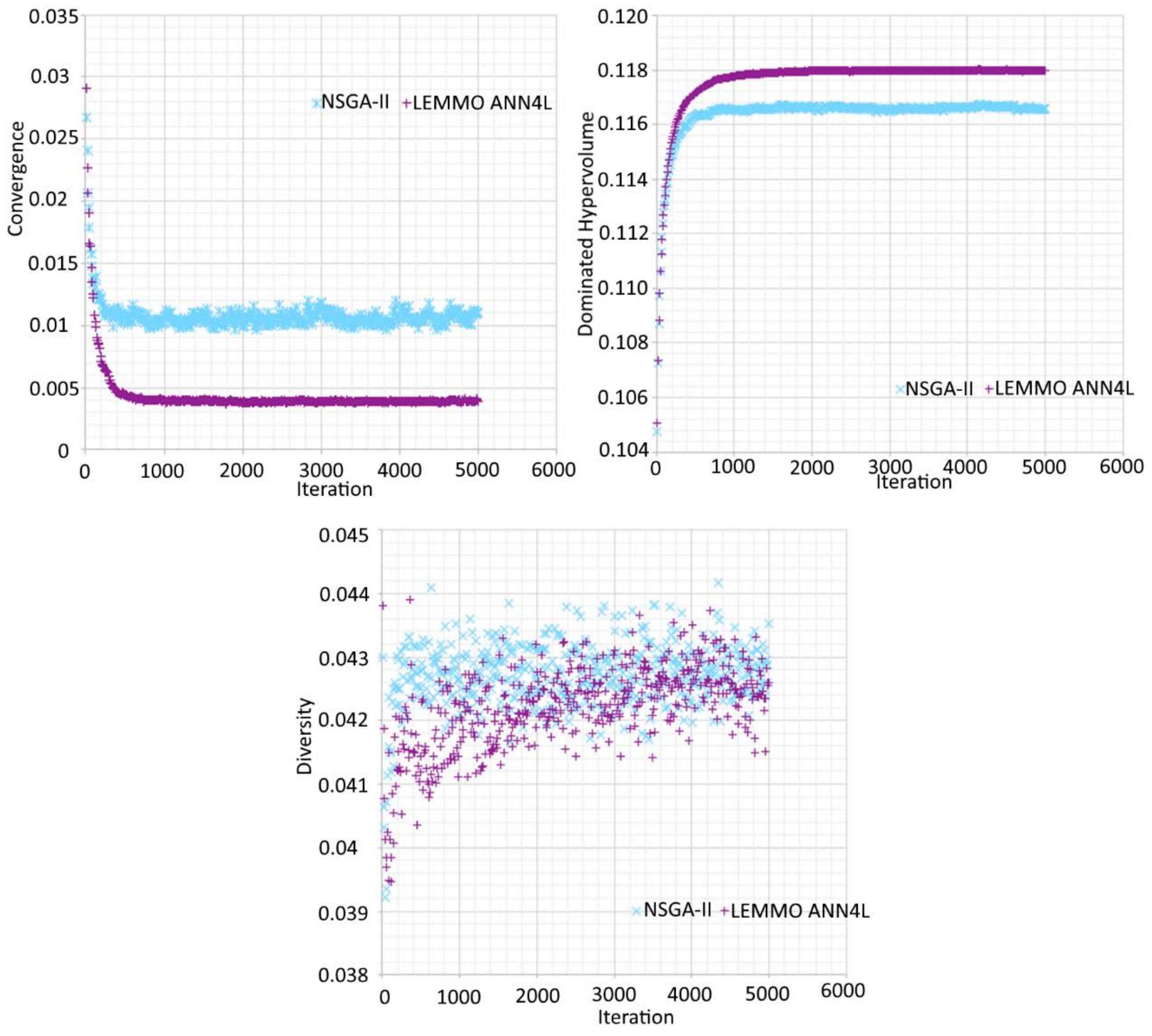
Figure 6. Averaged metrics for TLN



GOY

The analysis of the GOY test (see figure 7) shows that in terms of convergence and dominated hypervolume, LEMMO-ANN consistently out-performs the NSGA-II base algorithm. In terms of diversity, the metric appears to be extremely variable from iteration to iteration. However, the overall mean diversity for the NGSA-II base algorithm is 0.427, whereas the overall mean diversity for the LEMMO-ANN implementation is 0.422. So on average the NSGA-II base algorithm is very slightly out-performing the LEMMO implementation in terms of diversity on this particular problem.

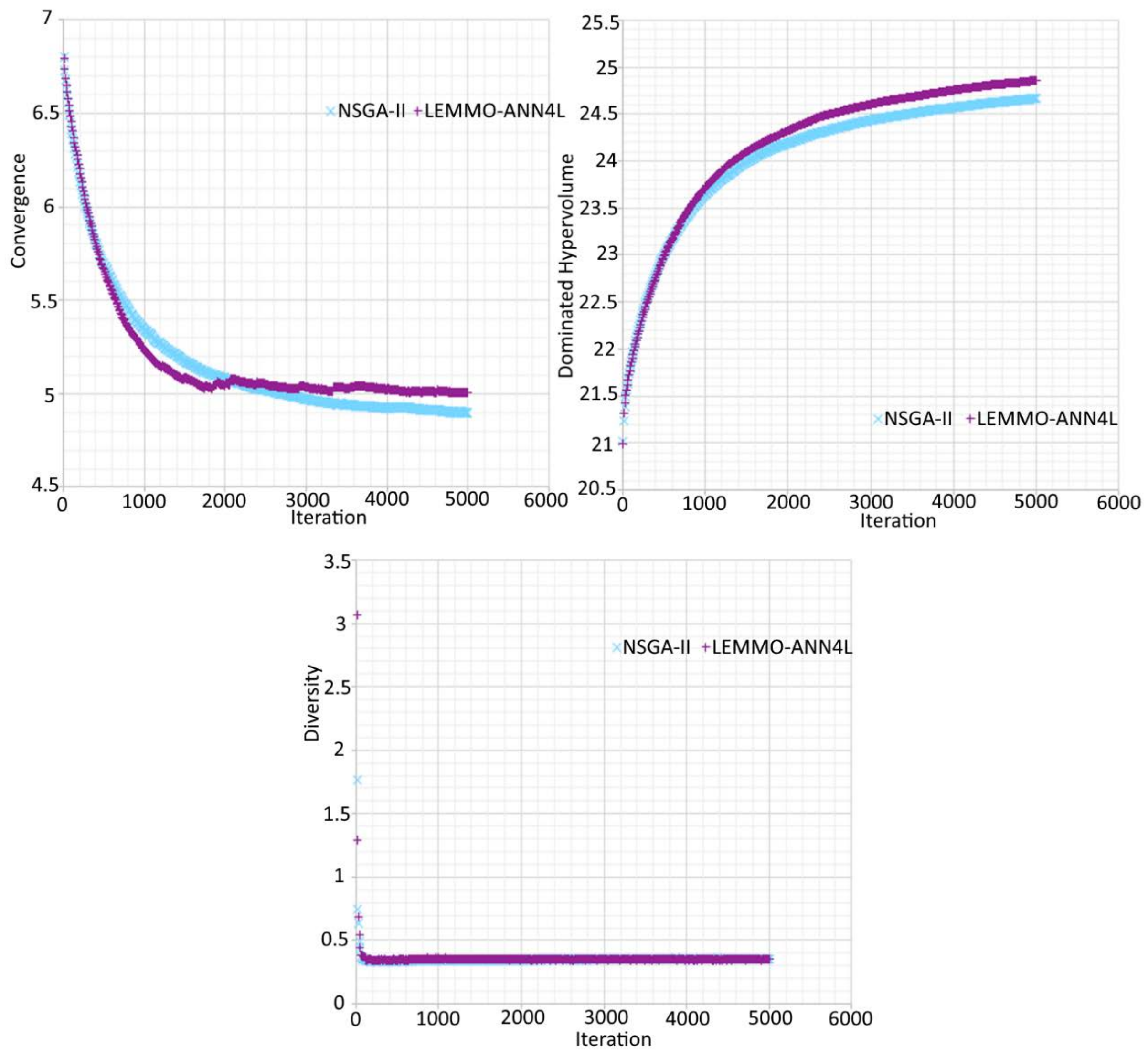
Figure 7. Averaged metrics for GOY



MOD

The analysis of MOD (see figure 8) shows that for convergence, the NSGA-II base algorithm out-performs LEMMO-ANN. In terms of diversity it appears that NSGA-II algorithm has very slightly higher diversity, followed by ANN LEMMO. However, in terms of dominated hypervolume LEMMO-ANN demonstrates improved results. This suggests that LEMMO-ANN is achieving a better estimation of the Pareto front, but with solutions that differ from NSGA-II's.

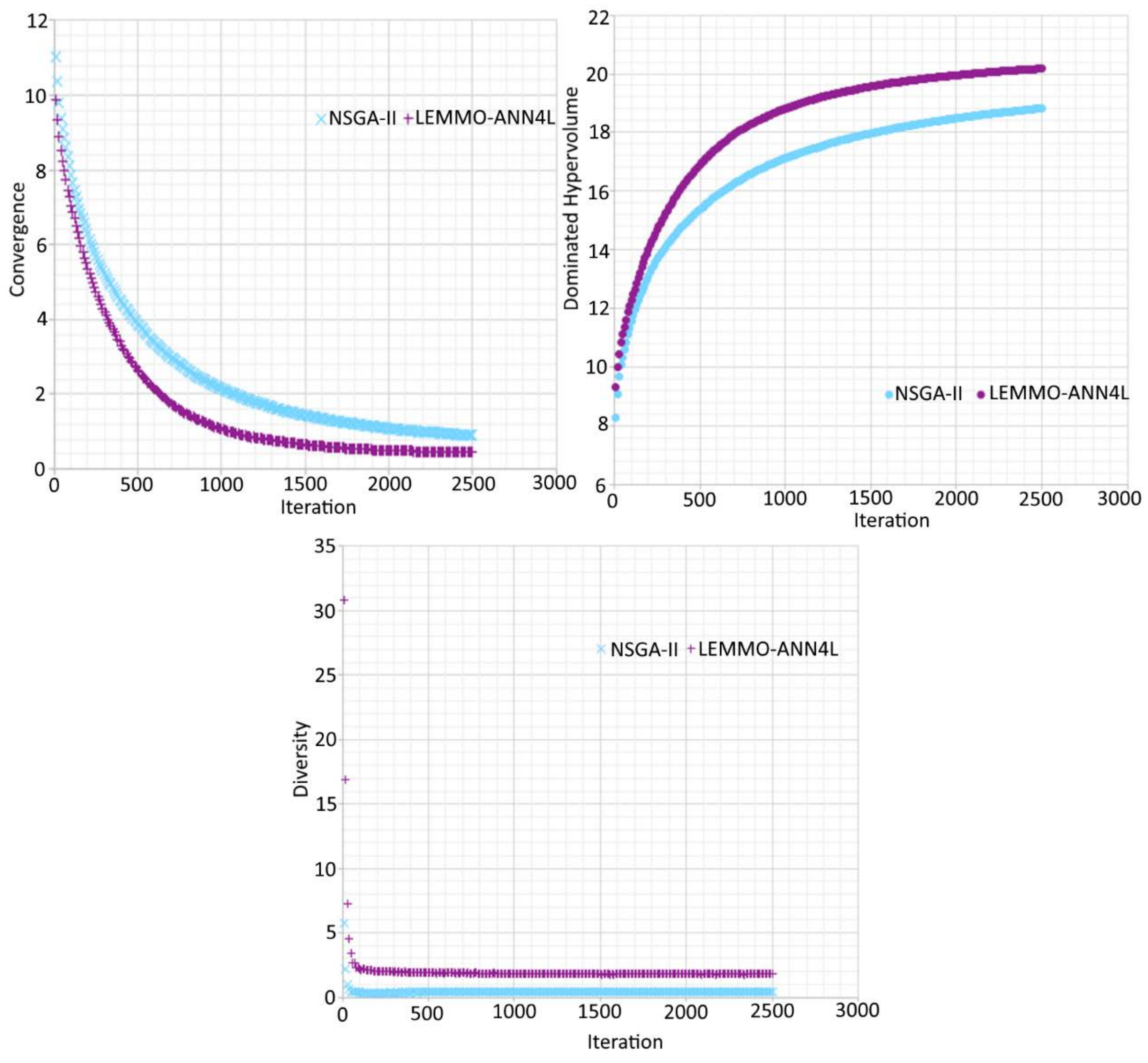
Figure 8. Averaged metrics for MOD



BIN

In the analysis of the metrics for the BIN WDS test problem (see figure 9) it is clear to see that LEMMO-ANN consistently out-performs NSGA-II in all cases.

Figure 9. Averaged metrics for BIN



It can be seen from the presented results here that the NSGA-II (Deb et al., 2002) algorithm converges well on test problems, approaching the best known Pareto fronts that are being used for comparison (Wang et al., 2014). . The LEMMO-ANN approach can be seen to achieve better convergence towards the same best-known Pareto fronts when using the same number of objective function evaluations. Additionally, it can be seen that the LEMMO-ANN approach achieves equivalent (to the NSGA-II base algorithm) or better convergence to the best-known Pareto fronts in fewer iterations (and thus fewer objective function evaluations). It is also entirely possible that further tuning the ANN structure, training strategy, and solution generation technique, could further improve the algorithms results in terms of how quickly they approach the optimal Pareto front, and how closely they match it.

Logically, no accuracy is lost through this process. The LEMMO-ANN algorithm integrates into the NSGA-II algorithm in such a way that if a LEMMO iteration produces only very poor solutions to the problem, they will not enter the population. The only anticipatable negative effect is that if the neural network cannot model the complexities of the problem well, it could bias the algorithm towards convergence to a local optimum.

Additionally, extra time taken to run a LEMMO iteration versus running a full iteration is negligible in this application, meaning that it is very cheap in terms of computational demand to use this technique to improve the results of the NSGA-II algorithm, for applications with a very computationally demanding objective function.

Conclusion

Within this article, an optimisation algorithm is suggested for the design of water distribution systems. This model is developed based on the LEMMO (Jourdan et al., 2005, 2004) algorithm, modified to use artificial neural networks as meta-heuristics. The algorithm modifies the existing water distribution system design attempting to optimise this with regard to two objectives, network resilience and capital expenditure. The results of this optimisation are then compared against the results of the same optimisation performed with standard NSGA-II (Deb et al., 2002) in terms of diversity, convergence and dominated hypervolume. The results compared overall favourably with the pareto fronts generated by NSGA-II, with less computational effort invested. Given the same amount of computational effort, the results improved upon the NSGA-II results. The conclusion, therefore, is that the LEMMO approach used with NSGA-II performs well with ANN meta-models as in the LEMMO-ANN approach. It generally improves the results compared to a standard NSGA-II base algorithm and achieves comparable results in fewer iterations. The caveat is that the ANN used must be structured and trained well enough that it will approximate the testing function well, otherwise it could potentially bias the algorithm towards local optima.

References:

Bach, P.M., Rauch, W., Mikkelsen, P.S., McCarthy, D.T., Deletic, A., 2014. A critical review of integrated urban water modelling - Urban drainage and beyond. Environ. Model. Softw. 54, 88–107. https://doi.org/10.1016/j.envsoft.2013.12.018

Behzadian, K., Kapelan, Z., Savić, D.A., Ardeshir, A., 2009. Stochastic Sampling Design using Multiobjective Genetic Algorithm and Adaptive Neural Networks. Environ. Model. Softw. 24, 530–541.

Bekele, E.G., Nicklow, J., 2007. Multi-objective automatic calibration of SWAT using NSGA-II. J. Hydrol. 341, 165–176.

Chester, D.L., 1990. Why Two Hidden Layers are Better than One. IJCNN-90-WASH-DC 1, 265–366.

Cybenko, G., 1989. Approximation by superpositions of a sigmoidal function. Math. Control Signals Syst. 2, 303–314.

Deb, K., Agrawal, S., Pratab, A., Meyarivan, T., 2000. A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II, in: et Al., M.S. (Ed.), . Springer, Paris, France, pp. 849–858.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6, 182–197.

di Pierro, F., Khu, S.-T., Savić, D., Berardi, L., 2009. Efficient multi-objective optimal design of water distribution networks on a budget of simulations using hybrid algorithms. Environ. Model. Softw. 24, 202–213. https://doi.org/10.1016/j.envsoft.2008.06.008

Djordjević, S., Prodanović, D., Maksimović, C., 1999. An approach to simulation of dual drainage. Water Sci. Technol. 39, 95–103.

Foord, M., Muirhead, C., 2009. IronPython in Action. Manning Publications Co., Greenwich, CT, USA.

Fortin, F.-A., De Rainville, F.-M., Gardner, M.-A., Parizeau, M., Gagné, C., 2012. DEAP: Evolutionary Algorithms Made Easy. J. Mach. Learn. Res. 13, 2171–2175.

Garcìa, L., Barreiro-Gomez, J., Escobar, E., Téllez, D., Quijano, N., Ocampo-Martinez, C., 2015. Modeling and Real-Time Control of Urban Drainage Systems: A Review. Adv. Water Resour. https://doi.org/10.1016/j.advwatres.2015.08.007

Hammond, M., Chen, A.S., Batica, J., Butler, D., Djordjević, S., Gourbesville, P., Manojlović, N., Mark, O., Veerbeek, W., 2018. A new flood risk assessment framework for evaluating the effectiveness of policies to improve urban flood resilience. Urban Water J. 15, 427–436. https://doi.org/10.1080/1573062X.2018.1508598

Hartman, E.J., Keeler, J.D., Kowalski, J.M., 1990. Layered neural networks with gaussian hidden units as universal approximations. Neural Comput. 2, 210–215.

Hornik, K., 1991. Approximation Capabilities of Mutlilayer Feedforward Networks. Neural Netw. 4, 251–257.

Hornik, K., Stinchcombe, M., White, H., 1989. Multilayer feedfordward networks are universal approximators. Neural Netw. 2, 359–366.

Igel, C., Hüsken, M., 2000. Improving the Rprop Learning Algorithm, in: Proceedings of the Second International Symposium on Neural Computation.

Jourdan, L., Corne, D., Savić, D., Walters, G., 2005. Preliminary Investigation of the `Learnable Evolution Model' for Faster/Better Multiobjective Water Systems Design. Evol. Multi-Criterion Optim. 841–855.

Jourdan, L., Corne, D., Savić, D., Walters, G., 2004. Hybridising Rule Induction and Multi-Objective Evolutionary Search for Optimising Water Distribution Systems, in: Hybrid Intelligent Systems, 2004. HIS '04. Fourth International Conference On. Presented at the Fourth International Conference on Hybrid Intelligent Systems, IEEE, pp. 434–439. https://doi.org/10.1109/ICHIS.2004.58

Kannan, S., Baskar, S., McCalley, J.D., Murugan, P., 2009. Application of NSGA-II Algorithm to Generation Expansion Planning. IEEE Trans. Power Syst. 24, 454–461.

Masters, T., 1993. Practical Neural Network Recipes in C++. Morgan Kauffman Publishers.

Michalski, R.S., Esposito, F., Saitta, L., 2000. Learning Evolution model: Evolutionary Processes Guided by Machine Learning. Mach. Learn. 38, 9–40.

Mohtar, W.H.M.W., Afan, H., El-Shafie, A., Bong, C.H.J., Ghani, A.A., 2018. Influence of bed deposit in the prediction of incipient sediment motion in sewers using artificial neural networks. Urban Water J. 15, 296–302. https://doi.org/10.1080/1573062X.2018.1455880

Park, J., Sandberg, I.W., 1991. Universal approximation using radial basis function networks. Neural Comput. 3, 246–257.

Prasad, T.D., Park, N.-S., 2004. Multiobjective Genetic Algorithms for Design of Water Distribution Networks. J. Water Resour. Plan. Manag. 130, 73–82. https://doi.org/10.1061/(ASCE)0733-9496(2004)130:1(73)

Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Riedmiller, M., Braun, H., 1993. A direct adaptive method for faster backpropagation learning: The rprop algorithm. Proc. IEEE Int. Conf. Neural Netw. 586–591.

Sayers, W., 2015. Artificial Intelligence Techniques for Flood Risk Management in Urban Environments (EngD). University of Exeter, Exeter, UK.

Sayers, W., Savić, D., Kapelan, Z., Kellagher, R., 2014. Artificial Intelligence Techniques for Flood Risk Management in Urban Environments, in: Procedia Engineering. Presented at the 12th International Conference on Computing and Control for the

Water Industry, Perugia, Italy, pp. 1505–1512. https://doi.org/10.1016/j.proeng.2014.02.165

Shirzad, A., 2017. Shortening the search time in optimization of water distribution networks. Urban Water J. 14, 1038–1044. https://doi.org/10.1080/1573062X.2017.1325502

Shishegar, S., Duchesne, S., Pelletier, G., 2018. Optimization methods applied to stormwater management problems: a review. Urban Water J. 15, 276–286. https://doi.org/10.1080/1573062X.2018.1439976

Souza, C., 2015. Accord Framework .NET.

Svozil, D., Kvasnička, V., Pospíchal, J., 1997. Introduction to multi-layer feed-forward neural networks. Chemom. Intell. Lab. Syst. 39, 43–62.

Wang, Q., Guidolin, M., Savic, D., Kapelan, Z., 2014. Two-Objective Design of Benchmark Problems of a Water Distribution System via MOEAs: Towards the Best-Known Approximation of the True Pareto Front. J. Water Resour. Plan. Manag. 141, 04014060. https://doi.org/10.1061/(ASCE)WR.1943-5452.0000460

Wang, Q.J., 1997. Using Genetic Algorithms to Optimise Model Parameters. Environ. Model. Softw. 27–34.

Webber, J.L., Gibson, M.J., Chen, A.S., Savic, D., Fu, G., Butler, D., 2018. Rapid assessment of surface-water flood-management options in urban catchments. Urban Water J. 15, 210–217. https://doi.org/10.1080/1573062X.2018.1424212

Wessing, S., 2010. HyperVolume. TU Dortmund University.

Wojtusiak, J., Michalski, R., 2006. The LEM3 Implementation of learnable evolution model and its testing on complex function optimization problems, in: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation. Presented at the 8th Annual Conference on Genetic and Evolutionary Computation, ACM, pp. 1281–1288. https://doi.org/10.1145/1143997.1144197

Woodward, M., 2012. The use of real options and multi-objective optimisation in flood risk management (PhD Thesis). University of Exeter, Exeter, UK.

Woodward, M., Gouldby, B., Kapelan, Z., Hames, D., 2013a. Multiobjective optimisation for improved management of flood risk. J. Water Resour. Plan. Manag. ASCE 2, 201–215.

Woodward, M., Kapelan, Z., Gouldby, B., 2013b. Adaptive Flood Risk management under Climate Change Uncertainty using Real Options and Optimisation. Risk Anal. 1, 75–92.

Zheng, J.-F., Jiao, J.-D., Zhang, S., Sun, L.-P., 2017. An optimization model for water quantity and quality integrated management of an urban lake in a water deficient city. Urban Water J. 14, 922–929. https://doi.org/10.1080/1573062X.2017.1301500

Zheng, Z., Li, J., Duan, P., 2019. Optimal chiller loading by improved artificial fish swarm algorithm for energy saving. Math. Comput. Simul., International Conference on Mathematical Modeling and Computational Methods in Science and Engineering 155, 227–243. https://doi.org/10.1016/j.matcom.2018.04.013

Zitzler, E., Thiele, L., 1998. Multiobjective Optimization Using Evolutionary Algorithms - A Comparative Study, in: Eiben, A.E. (Ed.), . Springer-Verlag, Amsterdam, pp. 292–301.

Table 1. Convergence metric example data 'A'

Example Data 'A'			**Example Data 'B'**		
X	**Y**	**Distances**	**X**	**Y**	**Distances**
7	0	2.00			
6	1	1.41	4.00	1.00	0.00
5	2	1.41			
3	3	1.00			
3	4	1.41	5.50	0.50	0.71
1	4	0.00			
Average Distance:		1.21	Average Distance:		0.35
Example Pareto Front Data					
X	**Y**	**'A' Distances**		**'B' Distances**	
5	0	1.41		0.71	
4	1	1.41		0.00	
3	2	1.00		1.41	
2	3	1.00		2.83	
1	4	0.00		4.24	
0	5	1.41		5.66	

Table 2. Test problem categories (the size of the search space given in the brackets)

Small	**Medium**	**Large**	**Very Large**
Two-reservoir network (TRN) (3.28×10^{7})	New York tunnel network (NYT) (1.93×10^{25})	Fossolo network (FOS) (7.25×10^{77})	Modena network (MOD) (1.32×10^{353})
Two-loop network (TLN) (1.48×10^{9})	Blacksburg network (BLA) (2.30×10^{26})	Pescara network (PES) (1.91×10^{110})	Balerma irrigation network (BIN) (1.00×10^{455})
BakRyan network (BAK) (2.36×10^{9})	Hanoi network (HAN) (2.87×10^{26})		Exeter network (EXN) (2.95×10^{590})
	GoYang network (GOY) (1.24×10^{27})		

Table 3. Test problem details

Problem	Water Sources	Decision Variables (Pipes)	Pipe Diameter Options	Search Space Size
TLN	1	8	14	1.48×10^{9}
GOY	1	30	8	1.24×10^{27}
MOD	4	317	13	1.32×10^{353}
BIN	4	454	10	1.00×10^{455}

Table 4. Test problem computational budget in original benchmarking

Problem	**Number of Evaluations**	**Group 1 Pop.**	**Group 2 Pop.**	**Group 3 Pop.**
TLN	100,000	40	80	160
GOY	600,000	60	120	240
MOD	2,000,000	200	400	800
BIN	2,000,000	200	400	800

Table 5. Contribution to best-known Pareto front from NSGA-II (Wang et al., 2014)

Problem	**Group 1 Contribution**	**Group 2 Contribution**	**Group 3 Contribution**
TLN	54	74	77
GOY	4	23	31
MOD	71	61	26
BIN	8	67	179

Table 6. Percentage contribution to the best-known Pareto front from NSGA-II in percentages (Wang et al., 2014)

Problem	**Total Solutions in Best-Known Pareto front**	**Percent Discovered by NSGA-II (%)**
TLN	77	100
GOY	67	43.3
MOD	196	57.7
BIN	265	72.5

Figure 1. Initial neural network structure with six input nodes

Figure 2. Graphical representation of the neural network structure with ten input nodes (for illustration only - test-problems and real problems should have more inputs).

Figure 3. Implementation of LEMMO-ANN and NSGA2 Algorithm (simplified diagram)

Figure 4. Convergence metric example data

Figure 5. Dominated Hypervolume example, shaded area represents dominated volume from the single red reference point, to the blue line.

Figure 6. Averaged metrics for TLN

Figure 7. Averaged metrics for GOY

Figure 8. Averaged metrics for MOD

Figure 9. Averaged metrics for BIN