NEMO: Neural Enhancement for Multiobjective Optimization

Aaron Garrett, Gerry Dozier, and Kalyanmoy Deb

***Abstract*— In this paper, a neural network approach is presented to expand the Pareto-optimal front for multiobjective optimization problems. The network is trained using results obtained from the nondominated sorting genetic algorithm (NSGA-II) on a set of well-known benchmark multiobjective problems. Its performance is evaluated against NSGA-II, and the neural network is shown to perform extremely well. Using the same number of function evaluations, the neural network produces many times more non-dominated solutions than NSGA-II.**

I. Introduction

Many optimization problems involve multiple, often conflicting, objectives. These multiobjective optimization problems are often much more difficult to solve than single-objective problems [1]. Typically, solutions to such problems are actually sets of solutions, each of which represents a particular trade-off for the objectives in question [1]. The more elements in such a set, the more options that are available as possible solutions. Increasing the size of this solution set has proven to be an extremely difficult problem [11].

Many multiobjective optimization techniques have been developed over the years, and, most recently, evolutionary computation approaches have been applied with great success [2]. Approaches like the vector evaluated genetic algorithm [2], the multiobjective particle swarm optimizer [2], the non-dominated sorting genetic algorithm [3], [9], and ParEGO [6] have all been shown to be effective. However, each of these approaches only produce a small number of nondominated solutions in a given number of function evaluations. In this paper, a neural enhancement process is described that can, in essence, learn the areas where nondominated solutions are found. In this way, many such solutions can be generated without the need for a large number of function evaluations.

II. Background

A. Multiobjective Optimization

A multiobjective optimization problem is defined as follows:

$$\text{Minimize} \quad [f_1(\vec{x}), f_2(\vec{x}), \cdots, f_k(\vec{x})]$$

subject to the m inequality constraints

$$g_i(\vec{x}) \leq 0 \quad i = 1, 2, \ldots, m$$

and the p equality constraints

$$h_i(\vec{x}) = 0 \quad i = 1, 2, \ldots, p$$

where k is the number of objective functions and $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$. The vector $\vec{x} = [x_1, x_2, \ldots, x_n]$ is referred to as the vector of *decision variables*. We wish to find the values $x_1^*, x_2^*, \ldots, x_n^*$ that yield the optimum values for all the objective functions.

1) Pareto Optimality: For multiobjective optimization problems, solutions actually represent trade-offs between some objectives in favor of others. For that reason, there is no clear way to define what is meant by a "good" solution. Instead, another type of metric is used, known as *Pareto optimality* [1]. Using this metric, a solution s_1 is said to *dominate* another solution s_2 if s_1 is no worse than s_2 in any objective and if s_1 is strictly better than s_2 in at least one objective.

The set of all globally nondominated solutions for a particular multiobjective problem is referred to as the *Pareto optimal set* [1]. The image of the Pareto optimal set under the objective functions is called the *Pareto optimal frontier* (or often just the *Pareto-optimal front*) [1]. Finally, since the global Pareto optimal set is often unknown, most multiobjective optimization approaches actually produce a *Pareto set approximation* [11].

2) NSGA-II: The nondominated sorting genetic algorithm (NSGA) [9] was first introduced in 1994 by Srinivas and Deb as an evolutionary multiobjective optimizer. The NSGA algorithm operated like a simple genetic algorithm except for the selection mechanism. There, the individuals were ranked according to Pareto preference using multiple passes(i.e., the first pass ranks all truly nondominated points, the second pass ranks all nondominated points from the remainder, etc.). In each pass, the nondominated points are assigned a dummy fitness value and points with the same fitness value undergo fitness sharing.

In 2002, Deb et al introduced an improvement to NSGA that they termed NSGA-II [3]. NSGA-II uses a sorting routine that runs in $O(MN^2)$ time, rather than $O(MN^3)$ (where M is the number of objectives and N is the population size). This routine is made possible by the use of a new domination operator that relies on the idea of *crowding distance* instead of using fitness sharing [4]. This crowding distance ensures that the solutions adequately fill the objective space. Finally, NSGA-II uses elitism, which allows good individuals to continue to survive and reproduce.

B. General Regression Neural Networks

First introduced by Specht in 1991 [8], general regression neural networks (GRNNs) can be viewed as an extension

Aaron Garrett is with the Mathematical, Computing, and Information Sciences Department at Jacksonville State University, Jacksonville, AL. (email: agarrett@jsu.edu).

Gerry Dozier is the director of the Applied Computational Intelligence Laboratory at Auburn University, Auburn, AL. (email: doziegv@auburn.edu)

Kalyanmoy Deb holds Shri Deva Raj Chair Professor at the Department of Mechanical Engineering, Indian Institute of Technology Kanpur, India. (email: deb@iitk.ac.in)

to the k-nearest neighbor approximation using a distance-weighted average and a global neighborhood. GRNNs are lazy learners [7] since they do not form a model of the training set, choosing instead to store all training instances for future reference. The general form for the output of a GRNN given n training inputs (x_i, y_i), $1 \leq i \leq n$, is

$$F(x) = \frac{\sum_{i=1}^{n} (y_i d(x, x_i))}{\sum_{i=1}^{n} d(x, x_i)}$$

where $d(\cdot)$ is a weighting function, x_i is a training input, and y_i is the desired output for input i.

In this paper, a Gaussian weighting function was used as described below:

$$d(x, y) = e^{\frac{-||x-y||^2}{2\sigma^2}}$$

In this equation, $||\cdot||$ represents Euclidean distance, and σ is a smoothing parameter that specifies the neighborhood size. For the GRNN, it is possible for a different smoothing parameter to be used for each training input, but this approach often leads to overfitting (as well as an explosion of parameter values that must be determined). In this work, a single value for σ was used for all training inputs. The particular value used for each problem is given in the Results section.

C. Related Work

Hatzakis and Wallace [5] used prediction in order to determine the next location to search on the Pareto optimal set in order to find new optimal solutions for time-dependent (i.e., dynamic) multiobjective optimization problems. They used a predictor based on an autoregressive model to predict the next "likely" area to find optimal solutions once the fitness landscape shifts. Their predictor determined two anchor points on the Pareto-optimal front using the time-series information of the previous anchor points. The authors mention that "[o]ne might also fit an analytic curve which describes the Pareto optimal set (the locus of the Pareto-optimal front in variable space) and subsequently forecast changes in the parameter values of this curve, instead of using specific points".

In similar fashion, Knowles [6] developed ParEGO, which uses efficient global optimization (EGO) as a learning system to produce a model of the fitness landscape. In this work, the system attempted to learn the model of an expensive multiobjective function given some points on the Pareto-optimal front. This allowed the use of the model to find minima, which can then be checked using the expensive actual multiobjective function. The results were compared against those produced by NSGA-II on eight different test functions, and ParEGO was shown to perform extremely well, even though only 250 function evaluations were used.

The work presented in this paper is an extension of Yapicioglu et al's work from [10]. It differs from the previously mentioned approaches in that prediction is used to model the Pareto optimal set using a subset of the decision variables. In this way, the most promising regions of the optimal set can be explored in order to more quickly discover nondominated solutions. Rather than being concerned solely with dynamic or expensive multiobjective problems, this approach is applicable for all multiobjective problems and can be used with any pre-existing set of nondominated solutions, no matter how they are generated.

III. Methodology

The neural enhancement approach, which is called NEMO (neural enhancement for multiobjective optimization), was tested on a set of four benchmark problems from the multiobjective optimization literature, as well as a real-world multiobjective optimization problem. The results were compared against the performance of NSGA-II on the same problems using the same number of function evaluations. As a metric, the ratio of NEMO's solutions versus NSGA-II's solutions (herein denoted the *yield ratio*) was calculated for each test problem. The following subsections describe the experimental setup in more detail.

A. Test Suite

In order to provide a thorough test of the neural enhancement system, a subset of four multiobjective benchmark problems was extracted from [6]. These problems were DTLZ1a, DTLZ2a, DTLZ4a, and OKA2. Finally, the semi-desirable facility location problem (SDFLP) was included from [10] (where it was cited as test case 1.1).

1) DTLZ1a: The DTLZ1a test function makes use of six decision variables to minimize two objective functions, defined as follows:

$$\begin{aligned}
f_1 &= \frac{1}{2} x_1 (1 + g) \\
f_2 &= \frac{1}{2} (1 - x_1)(1 + g) \\
g &= 100 \left[5 + \sum_{i=2}^{6} \left((x_i - 0.5)^2 - \cos(2\pi(x_i - 0.5)) \right) \right] \\
x_i &\in [0, 1], \quad i \in \{1, \ldots, 6\}
\end{aligned}$$

The Pareto optimal set consists of all decision variables set to 0.5 except the first value, which should come from $[0, 1]$.

2) DTLZ2a: The DTLZ2a test function uses eight decision variables to minimize three objective function and is defined as follows:

$$\begin{aligned}
f_1 &= (1 + g) \cos(x_1 \frac{\pi}{2}) \cos(x_2 \frac{\pi}{2}) \\
f_2 &= (1 + g) \cos(x_1 \frac{\pi}{2}) \sin(x_2 \frac{\pi}{2}) \\
f_3 &= \sin(x_1 \frac{\pi}{2}) \\
g &= \sum_{i=3}^{8} (x_i - 0.5)^2 \\
x_i &\in [0, 1], \quad i \in \{1, \ldots, 8\}
\end{aligned}$$

The Pareto-optimal front for this function is one-eighth of a sphere centered at the origin with a radius of 1. The Pareto optimal set consists of all decision variables set to 0.5 except the first value, which should come from $[0, 1]$.

3) DTLZ4a: The DTLZ4a test function uses eight decision variables to minimize three objective function and is defined as follows:

$$
\begin{aligned}
f_1 &= (1+g)\cos(x_1^{100}\frac{\pi}{2})\cos(x_2^{100}\frac{\pi}{2}) \\
f_2 &= (1+g)\cos(x_1^{100}\frac{\pi}{2})\sin(x_2^{100}\frac{\pi}{2}) \\
f_3 &= \sin(x_1^{100}\frac{\pi}{2}) \\
g &= \sum_{i=3}^{8}(x_i - 0.5)^2 \\
x_i &\in [0,1], \quad i \in \{1,\ldots,8\}
\end{aligned}
$$

As with DTLZ1a, the Pareto-optimal front for this function is one-eighth of a sphere centered at the origin with a radius of 1. The Pareto optimal set consists of all decision variables set to 0.5 except the first value, which should come from $[0, 1]$.

4) OKA2: The OKA2 test function uses three decision variables to minimize two objective function and is defined as follows:

$$
\begin{aligned}
f_1 &= x_1 \\
f_2 &= 1 - \frac{1}{4\pi^2}(x_1+\pi)^2 + |x_2 - 5\cos(x_1)|^{\frac{1}{3}} \\
&\quad + |x_3 - 5\sin(x_1)|^{\frac{1}{3}} \\
x_1 &\in [-\pi, \pi] \\
x_2, x_3 &\in [-5, 5]
\end{aligned}
$$

The Pareto optimal set lies on a 3D spiral curve.

5) SDFLP: The semi-desirable facility location problem deals with positioning a facility (such as an airport or landfill) that is necessary for members of the community. However, the facility itself produces an undesirable by-product (such as traffic or pollution) that the community does not desire. The balance of desirable and undesirable effects leads to the multiobjective problem.

$$
\begin{aligned}
f_1 &= \sum_{i=1}^{7}(w_{1i}d(\vec{x},\vec{a_i})) \\
f_2 &= \sum_{i=1}^{7}v_i(\vec{x},\vec{a_i}) \\
v_i(\vec{x},\vec{a_i}) &= \begin{cases} 200 & \text{if } w_{2i}d(\vec{x},\vec{a_i}) < 10 \\ 200 - w_{2i}d(\vec{x},\vec{a_i}) & \text{if } 10 \le w_{2i}d(\vec{x},\vec{a_i}) < 30 \\ 0 & \text{if } 30 \le w_{2i}d(\vec{x},\vec{a_i}) \end{cases} \\
x_1, x_2 &\in [-20, 40] \\
\vec{a} &= ((5,20),(18,8),(22,16),(14,17), \\
&\quad (7,2),(5,15),(12,4)) \\
\vec{w_1} &= (5,7,2,3,6,1,5) \\
\vec{w_2} &= (1,1,1,1,1,1,1)
\end{aligned}
$$

Since the SDFLP is a real-world problem, its Pareto optimal set is unknown.

B. NEMO: The Neural Enhancer

To perform the neural enhancement for each test problem, a GRNN was created using a subset of the decision variables in the Pareto set approximation to predict the values for the remaining decision variables in the set. In essence, the GRNN is used to learn the mapping, in the Pareto set approximation, from some subset of the decision variables to the remaining decision variables.

For instance, given a multiobjective optimization problem with 5 decision variables, $d_1, \ldots, d_5$, NEMO first partitions the decision variables into two sets, I and O. For instance, suppose that I contains d_1 and d_3, while O contains d_2, d_4, and d_5. Then, a GRNN is created using elements of I as inputs and producing elements of O as outputs. The GRNN is trained using the elements from the original Pareto set approximation (as found by NSGA-II). Then, values, call them i_1 and i_3 are generated uniformly from the range of each of the elements of I. These values are then passed into the GRNN to produce the corresponding values o_2, o_4, and o_5 in set O. Finally, the generated solution $< i_1, o_2, i_3, o_4, o_5 >$ is passed to the multiobjective function to discover the objective values associated with it. These values determine whether the generated solution is nondominated.

A steady-state genetic algorithm (GA) was used to evolve the subset of decision variables to be used as inputs to the GRNN, as well as the value of the GRNN's sigma parameter. The parameters of the GA were chosen arbitrarily – population size of 100, uniform crossover, bit-flip mutation on the binary segment of the chromosome (i.e., the inclusion/exclusion of each decision variable), and Gaussian mutation on the real-coded segment of the chromosome (i.e., the sigma parameter). The crossover usage rate was set to 1.0, the mutation rates for both types of mutations were set to 0.1, and the mutation range for the Gaussian mutation was set to 1.0.

To evaluate each candidate solution, a GRNN was constructed using the appropriate decision variables and sigma value. The sigma value was taken from the interval [0.01, 20.0]. The GRNN was then trained using the Pareto optimal set for the given test problem. Then, the neural network was presented with 10 randomly generated "inputs" and was asked to predict the corresponding "outputs", which together constituted a possible point in the Pareto optimal set. Each point was then evaluated using the given test problem to produce values for the objective functions. Those values were compared to the values in the Pareto-optimal front to determine whether the point should be added, and, if so, whether it replaced any existing points in the Pareto-optimal front. The fitness for the candidate solution (i.e., the GRNN parameters) was then taken to be

$$fitness = 2 \cdot replaced + added.$$

Here, *replaced* represents the number of solutions in the original Pareto-optimal front that were dominated by the 10 solutions generated by the neural network. Likewise, *added* represents the number of solutions generated by the neural

network that were nondominated in the original Pareto-optimal front. The intention of this fitness measure was to more heavily favor neural networks capable of producing nondominated solutions in the existing Pareto-optimal front. The GA was given a maximum of 1500 function evaluations in order to find the best parameters for the GRNN.

After the appropriate parameters were found, the GRNN was used to predict 10000 values in the Pareto optimal set. To accomplish this, equally spaced points were chosen along the entire range of the GRNN's input dimensions and were passed to the neural refiner to determine the values along the remaining dimensions. (The distance between points was chosen so that no more than 10000 values would be created.) Each value in the predicted Pareto optimal set was then evaluated using the multiobjective test function. The results are described below.

IV. Results

The following subsections describe the results of applying NEMO to the test suite. These results are summarized in Table I. In this table, the "NEMO Added" column displays the number of solutions that were added by NEMO to the original Pareto-optimal front (produced by NSGA-II). Similarly, the "NEMO Replaced" column holds the number of NEMO solutions that replaced at least one solution from the original Pareto-optimal front. Finally, the "Yield Ratio" determines the relative effectiveness of NEMO versus NSGA-II, as described previously.

A. DTLZ1a

For the DTLZ1a problem, the Pareto-optimal front found using NSGA-II, consisting of 200 points, is shown in red in Figure 1. The neural enhancer was created using the first decision variable as input to predict the remaining 5 variables. The sigma value was set to 8.04. The results of generating 10000 points using the neural refiner are shown in green in Figure 1. Here, we see that the GRNN created 9999 non-dominated solutions in the predicted Pareto-optimal front (by which we mean that only one of the predicted points was dominated by other predicted points). These predicted values were merged with the original values found by NSGA-II, allowing the predicted values to dominate or be dominated by the original values, which yielded a Pareto-optimal front consisting of 9638 points.

B. DTLZ2a

For the DTLZ2a problem, the Pareto-optimal front found using NSGA-II, consisting of 481 points, is shown in red in Figure 2. The neural enhancer was created using the first and second decision variables as input to predict the remaining 6 variables, using a sigma of 1.41. The results of generating 10000 points using the neural refiner are shown in green in Figure 2. There were 10000 non-dominated points in the final predicted Pareto-optimal front. When these points were merged with the original Pareto-optimal front, the final Pareto-optimal front consisted of 9572 points.

C. DTLZ4a

For the DTLZ4a problem, the Pareto-optimal front found using NSGA-II, consisting of 631 points, is shown in red in Figure 3. The neural enhancer was created using the second decision variable as input to predict the remaining 7 variables, using a sigma of 3.78. The results of generating 10000 points using the neural refiner are shown in green in Figure 3. There were 9999 non-dominated points in the final predicted Pareto-optimal front. Notice that the predicted values for the third objective all lie in the range [0.02715, 0.02745]. When the predicted and original values were merged, the final Pareto-optimal front consisted of 10626 points.

Although the neural refiner was able to generate over 15 times the number of nondominated points that NSGA-II produced, the spread of those points was not sufficient. It was believed that the failure on this problem was that the smoothing parameter (σ) for the GRNN was too large. A further experiment was conducted in which the smoothing parameter was constrained to the interval $[0.001, 0.00000001]$. The result of this approach, using a sigma of 0.000081, is shown in Figure 4. Here, it is clear that the coverage produced is much better than in Figure 3.

D. OKA2

For the OKA2 problem, the Pareto-optimal front found using NSGA-II, consisting of 15 points, is shown in red in Figure 5. The neural enhancer was created using the third decision variable as input to predict the first 2 variables, using a sigma of 0.01, which was the lower bound used by the GA. The results of generating 10000 points using the neural refiner are shown in green in Figure 5. There were 39 non-dominated points in the final predicted Pareto-optimal front.

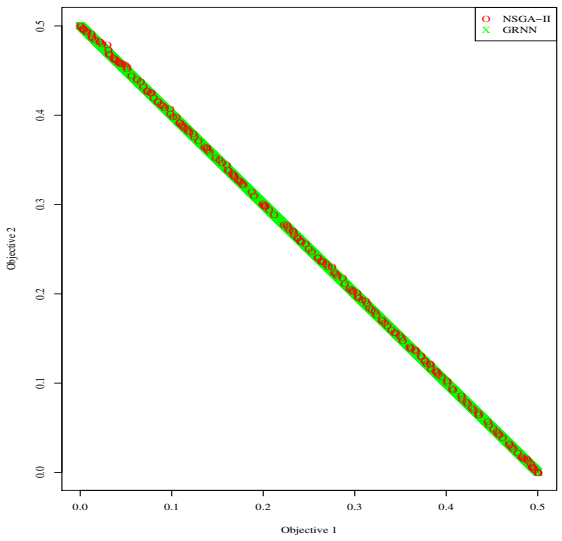

Fig. 1. Overlay of Pareto-optimal fronts for DTLZ1a from NSGA-II and GRNN

Problem	NSGA-II Front	NEMO Added	NEMO Replaced	Merged Front	Yield Ratio	σ
DTLZ1	200	6926	2592	9638	47.6	8.04
DTLZ2	481	7849	1486	9572	19.4	1.41
DTLZ4	631	9991	9	10626	15.9	3.78
OKA2	15	84	99	39	12.2	0.01
SDFLP	1259	616	279	2015	0.71	0.61

TABLE I
SUMMARY OF NEMO RESULTS

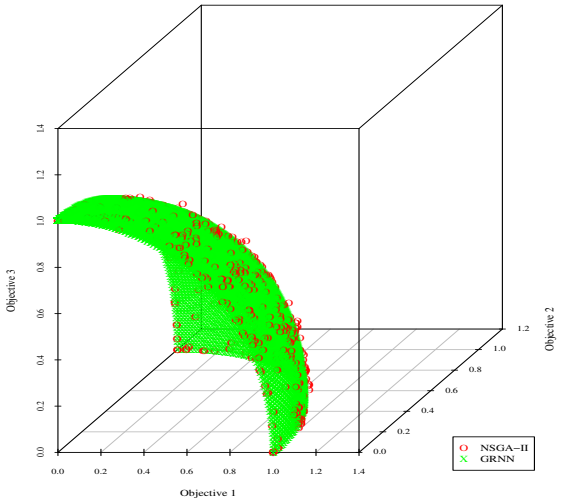

Fig. 2. Overlay of Pareto-optimal fronts for DTLZ2a from NSGA-II and GRNN

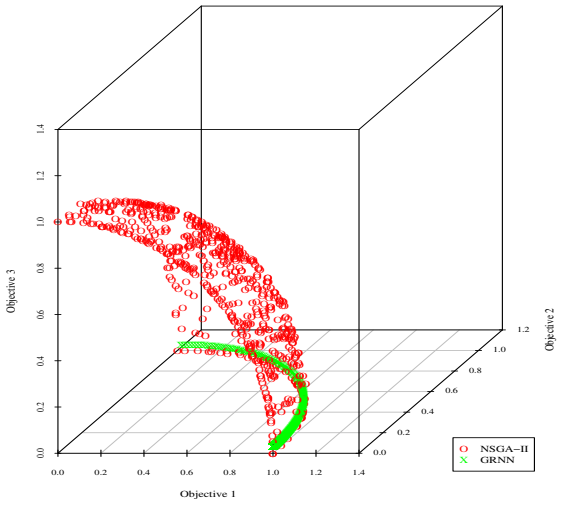

Fig. 3. Overlay of Pareto-optimal fronts for DTLZ4a from NSGA-II and GRNN

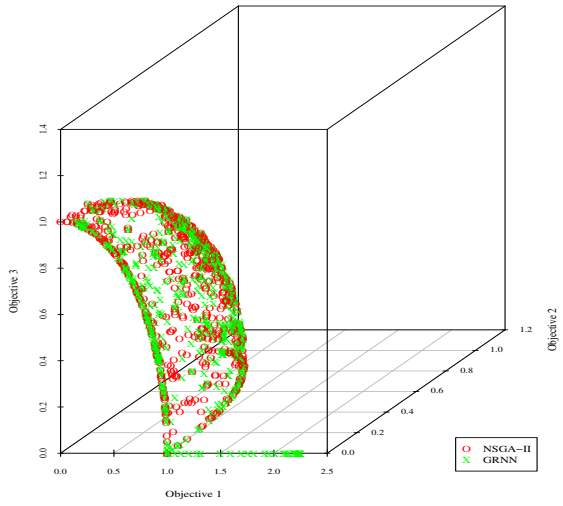

Fig. 4. Overlay of Pareto-optimal fronts for DTLZ4a from NSGA-II and GRNN using small σ value

After the predicted and original values were merged, the final Pareto-optimal front also consisted of 39 points.

E. SDFLP

For the SDFLP problem, the Pareto-optimal front found using NSGA-II, consisting of 1259 points, is shown in red in Figure 6. The neural enhancer was created using the second decision variable as input to predict the first, using a sigma of 0.61. The results of generating 10000 points using the neural refiner are shown in green in Figure 6. There were 3753 non-dominated points in the final predicted Pareto-optimal front. After merging the predicted and original values, the Pareto-optimal front consisted of 2015 points.

V. CONCLUSIONS AND FUTURE WORK

The neural enhancement approach presented here has been shown to be quite effective at expanding the Pareto optimal frontier for several benchmark multiobjective optimization problems. For each problem, the NEMO approach produced over 10 times more solutions than NSGA-II. However, the real-world problem tested proved more difficult. NEMO was only able to produce about 70% of the solutions that NSGA-

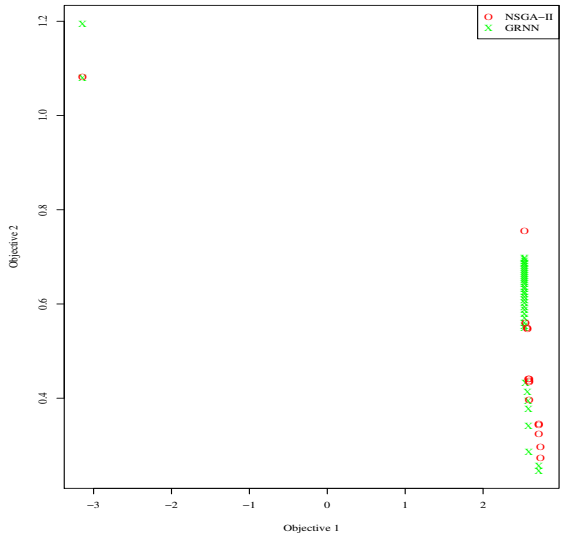

Fig. 5. Overlay of Pareto-optimal fronts for OKA2 from NSGA-II and GRNN

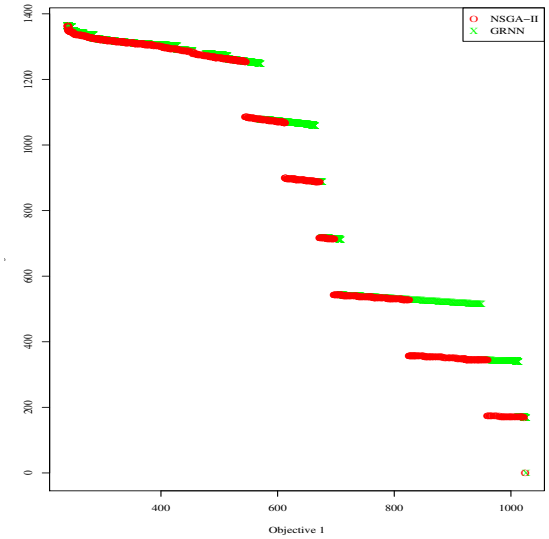

Fig. 6. Overlay of Pareto-optimal fronts for SDFLP from NSGA-II and GRNN

II produced. Even so, about 20% of those solutions replaced the solutions produced by NSGA-II.

Given the very promising results presented here, further work must be done to determine their statistical significance. An extension of this study can also be made for problems having more than three objectives. Additionally, work should be done to find a more appropriate training approach for the GRNN, either through a better-tuned evolutionary computation system or through a different training method such as leave-one-out training. In this way, an acceptable smoothing parameter can be found more quickly, which will provide greater performance for the neural enhancer. In addition to a more efficient training algorithm, a computational time comparison can be made between NEMO and NSGA-II.

REFERENCES

[1] C. A. C. Coello, "20 years of evolutionary multi-objective optimization: What has been done and what remains to be done," in *Computational Intelligence: Principles and Practice*, G. Y. Yen and D. B. Fogel, Eds. IEEE Computational Intelligence Society, 2006, pp. 73–88.

[2] C. A. C. Coello and M. Lechunga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proceedings of IEEE World Congress on Computational Intelligence*, 2002, pp. 1051–1056.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, apr 2002.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Reading, MA: Addison-Wesley Publishing Company, Inc., 1989.

[5] I. Hatzakis and D. Wallace, "Dynamic multi-objective optimization with evolutionary algorithms: A forward-looking approach," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO '06)*, 2006, pp. 1201–1208.

[6] J. Knowles, "ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, feb 2005.

[7] T. M. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[8] D. F. Specht, "A general regression neural network," *IEEE Transactions on Neural Networks*, vol. 2, no. 6, pp. 568–576, 1991.

[9] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[10] H. Yapicioglu, G. Dozier, and A. E. Smith, "Neural network enhancement of multiobjective evolutionary search," in *Proceedings of the 2006 Congress on Evolutionary Computation*, 2006.

[11] E. Zitzler, M. Laumanns, and S. Bleuler, "A tutorial on evolutionary multiobjective optimization," in *Workshop on Multiple Objective Metaheuristics (MOMH 2002)*. Berlin, Germany: Springer-Verlag, 2002.