Investigating Predator-Prey Algorithms for Multi-Objective Optimization

Kalyanmoy Deb and Udaya Bhaskara Rao N
Kanpur Genetic Algorithms Laboratory (KanGAL)
Department of Mechanical Engineering
Indian Institute of Technology Kanpur
Kanpur, PIN 208016, India
{deb,udaya}@iitk.ac.in
http://www.iitk.ac.in/kangal/

KanGAL Report Number 2005010

ABSTRACT

This paper describes the GA model using a new selection method inspired by predator-prey interactions. In this model, prey, which represents the decision space vector, will be placed on the vertices of a two-dimensional lattice. Predator, which deals with objective functions, will also be placed on the same lattice randomly. Basic algorithm proposed by Professor Hans-Paul Schwefel and reported in Laumanns et al. (1998) and the modifications on it are discussed here. Thereafter, we propose a number of modifications to the basic model and apply the final algorithm to standard two and three-objective optimization problems. The predator and prey approach is also used to find preferred Pareto-optimal solutions (preys) corresponding to user-supplied reference points (treated as predators). This study should encourage further use of predator-prey approaches to multi-objective optimization.

Keywords: Evolutionary algorithms; predator-prey algorithm; Evolutionary multi-objective optimization, reference point approach.

1. INTRODUCTION:

Over the past decade, evolutionary algorithms have been extensively used and applied to solve multi-objective optimization problems [1,2]. The main advantage of the evolutionary multi-objective optimization (EMO) procedures is that they can be used to find a set of Pareto-optimal solutions, instead of a single solution. In the presence of multiple trade-off solutions, (i) decision-makers and designers can make a better decision of choosing a solution [1,3] and (ii) designers can understand their problems better by deciphering salient inter-relationships among variables and objectives [4,5]. EMO methodologies are also being used for solving other kinds of optimization problems (such as constrained handling, reducing bloating in a genetic programming, and introducing adequate diversity thorough the use of an additional objective) [6,7].

Most EMO algorithms used a standard population-based EA with following three operators: (i) emphasizing non-dominated solutions for achieving convergence towards the Pareto-optimal front, (ii) emphasizing less-crowded solutions for achieving a well-diverse set of solutions, and (iii) an elite-preservation operator for making a faster and reliable convergence. Some of the popular EMO methodologies are elitist non-dominated sorting GA or NSGA-II [8], strength Pareto EA or SPEA2 [9] and others. Some of these algorithms are also available in commercial softwares (such as ISIGHT from Engeneous Inc. and mode FRONTIER from ESTICHO). Moreover, freely downloadable softwares are also available from various sites [10,11].

Although an extension of a standard EA procedure for multi-objective optimization is natural, Prof. Hans-Paul Schwefel of University of Dortmund, Germany had a slightly different and more natural concept in mind. In 1998, he and his coauthors proposed a predator-prey model for finding multiple Pareto-optimal solutions in a unique way [12]. In their proposed predator-prey algorithm (which we refer here as the `original model'), a new selection method was used which was inspired by predator-prey interactions. Here each prey represents one decision space vector, and each predator deals with one objective function. The algorithm imitates the natural phenomenon that a predator swallows the weakest prey, meaning that a predator eliminates the worst prey in its neighborhood, which corresponds to the worst value of the objective in which the predator specializes. To implement such an idea, they proposed a toroidal grid (shown in Figure 1), in which a prey is randomly initialized in every node and one (or more) predator per objective was placed randomly on nodes. Thereafter, every predator considers all preys (solutions) in the neighborhood and deletes the prey corresponding to the worst objective value. Then, a random prey from the neighborhood is chosen and mutated. The mutated child solution is then placed to the deleted prey. The predator then takes a random walk to one of its neighboring nodes. This procedure is continued for all predators one at a time. By the way of deleting worst solutions with respect to all objectives, the algorithm emphasizes the best solutions and preliminary simulation results have shown to take the initial random population towards the Pareto-optimal front [12].

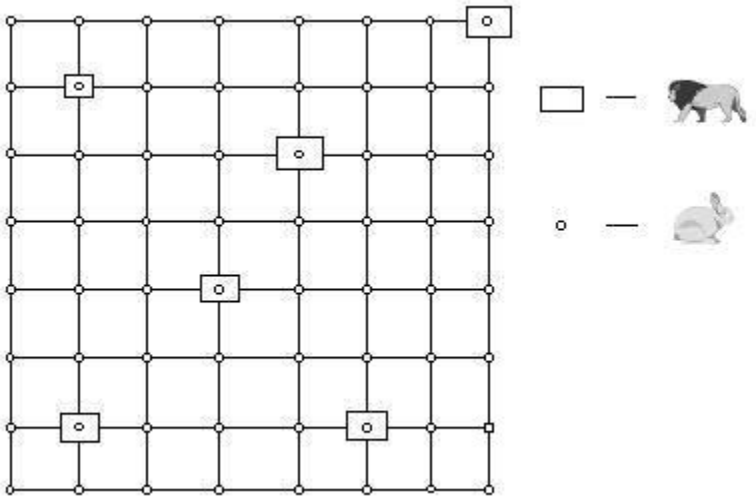

Figure 1: Placement of predators and preys on a toroidal grid (original model).

In this paper, we investigate the potential of such a predator-prey model in multi-objective optimization problem-solving by first outlining the reported extensions of the original model and then suggesting a number of new methodologies for an efficient solution of multi-objective optimization problems.

2. PAST METHODOLOGIES:

In the following, we describe the original model of Laumanns et al. [12] in a step-by-step manner:

Step 1: Initialize set of preys randomly between the variable limits.
Step 2: Place these preys on the vertices of undirected connected graph.
Step 3: Place predators randomly on the vertices of the graph.
Step 4: Assign each predator with one objective function in a manner so that each objective is assigned to at least one predator.
Step 5: Evaluate preys around each predator and select the worst prey. (For example, in a minimization problem, the worst prey will be the one which is having the largest value of the objective function which was assigned to that predator.)
Step 6: The selected preys will be swallowed by the predators, meaning that the worst prey will be deleted and will be replaced by an offspring.
Step 7: Create an offspring by mutating a randomly picked prey around the worst prey which was chosen by the predator.
Step 8: Then predators will now take a random walk to the vertex which is a neighbor of the current position of the predator.
Step 9: This completes one generation of the predator-prey algorithm. Repeat Steps 5 to 8 for the next generation.

With more generations, the prey population is hoped to reach near the true Pareto-optimal front. To illustrate the efficiency of this algorithm, we code the procedure and apply it to solve the following two-objective test problem:

Test-Problem 1 (TP1):
Minimize $f1(X) = x_1^2$,
Minimize $f2(X) = (1+x_2^2)/x_1^2$,
Subject to $\sqrt{0.1} \le x_1 \le 1$, $0 \le x_2 \le \sqrt{5}$.

Figure 2 shows the final population of 50 preys obtained after 200 generations. In this approach, one predator per objective function is used, thereby having only 2 predators in the entire grid of size 10 X 50. As can be seen from the figure, the convergence to the true Pareto-optimal front is not good.

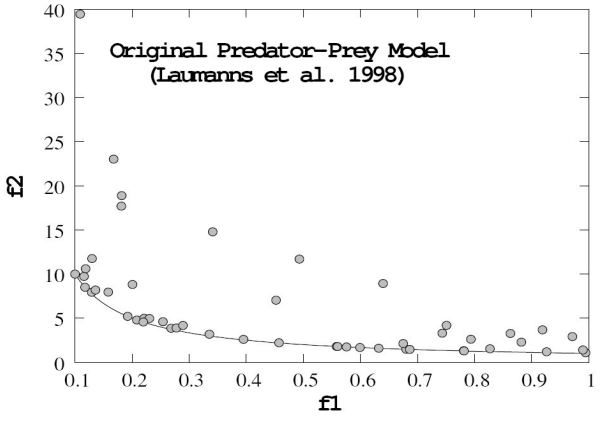

Figure 2: Original Predator-Prey model with one predator per objective function.

Next, we use 10 predators per objective, thereby using a total of 20 predators in the grid. Figure 3 shows the final population of 50 preys obtained after 200 generations. The convergence was still not good, because as the number of predators is increased, some well-converged preys may be deleted if placed in the neighborhood of some other predators.

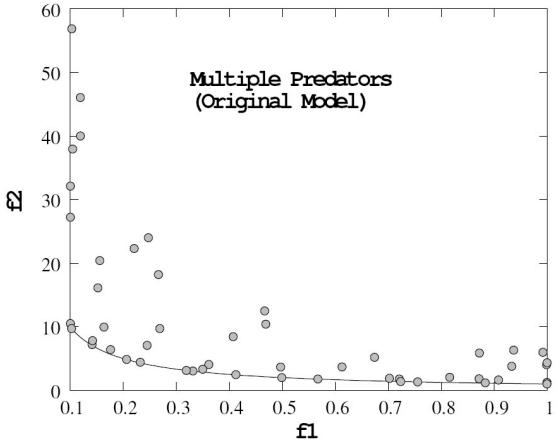

Figure 3: Original Predator-Prey model with 10 predators per objective.

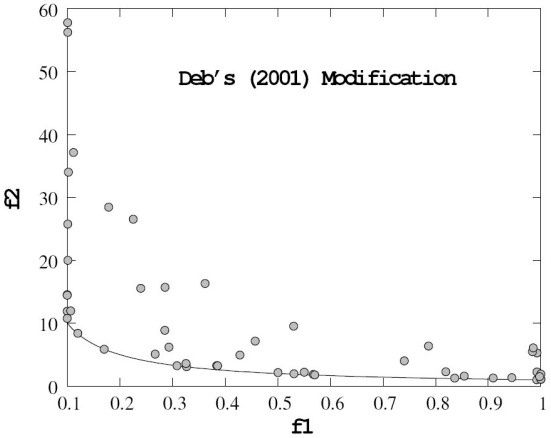

Figure 4: Modifications suggested in Deb (2001) with predators working with weighted sum of objectives. 10 predators per objective are chosen.

The first author suggested some modifications to the original algorithm in his 2001 book [1].

1. Predators assigned with weight vectors, instead of individual objective function. These predators will deal with the weighted sum of objective function values.
2. Offspring will be created by mutating the best prey around the worst prey, instead of random one.
3. Predator will move to the best prey position among its neighboring preys, instead of random move.

By these modifications the algorithm improved in such a manner that the convergence rate was good, but still elite-preservation was not introduced. Because of this as more predators are considered, it is likely that the some converged points may be taken out by other predators. Figure 4 shows the final population of 50 preys after 200 generations. 21 predators with weight vectors uniformly distributed between [0,1] to [1,0] are used.

Li [13] revised the original predator-prey algorithm and suggested that preys and predators be placed in the blocks of lattice (Figure 5), but need not completely fill the whole toroidal grid. All preys and predators will take random walks to their neighboring blocks which are empty. Predators will move faster than preys, meaning that the predator will move multiple times before a prey is moved. If the predator moves on to a prey position, that prey will be killed. All preys that get a chance to move will create offsprings. Since the preys are moved in a random fashion, the rate of convergence was found to be poor and the procedure took many iterations to come close to the Pareto-

optimal frontier. Another study [14] also used this idea to evolve neural network for a blast furnace modeling problem.

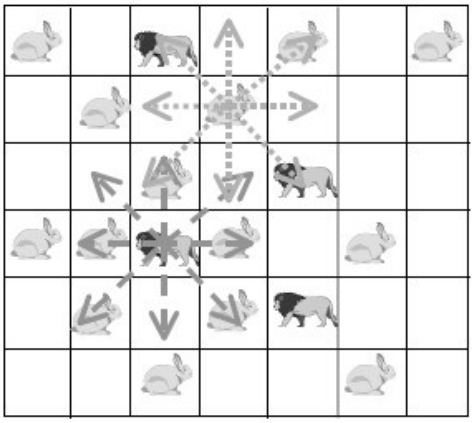

Figure 5: Predator-prey model proposed by Li [13].

3. PROPOSED MODIFICATIONS:

The present study modifies the original model systematically by introducing the following features one at a time:

1. Elite-preservation.
2. Recombination operator.
3. Diversity preservation mechanism.

We discuss the effect of each of the above features in the following subsections.

3.1 Introducing Elite-Preservation Operator:

Elite-preservation means preserving good solutions. Elitism can be implemented by comparing worst preys with newly created offspring. If only an offspring is found to be better than the worst prey, the worst prey will be replaced. The evaluation will be based on the *domination* criteria. If the offspring *weakly* dominates (best in at least one objective function) all existing preys, thereby meaning that no prey in the existing population strongly dominates (best in all objective functions) the offspring, then that offspring is fit for that population. If the created offspring is fit, then the worst prey will be replaced by the offspring. When the offspring is not found to be fit, the worst prey will remain in the population and the predator will take a random walk. By this way, we can introduce elitism. The random move of the predator among the entire vertices helps in maintaining the diversity somewhat. This situation happens only when the neighboring worst prey is better than the created offspring, thereby indicating that the region covered by this predator is good and a move by the predator may help find a region which may require finding better solutions. After creating the offspring, predator moves to the worst prey position instead of moving to the best prey position, as suggested by Deb [1]. This change is found to produce better results.

Figure 6 shows 50 preys obtained after 200 generations of this proposed elite-preservation procedure. Once again 21 predators with weight vectors uniformly

distributed between [0,1] to [1,0] are used. The figure shows that a much better converged set of solutions is obtained by the modified procedure.

3.2 Faster Convergence Using a Recombination Operation:

Crossover and mutation are introduced into the above modified predator-prey algorithm. Two offspring are created by applying a crossover between the best and the second best solutions around the worst prey. Thereafter, a mutation operator is applied on one of the selected offsprings at random. Figure 7 shows 50 preys after 200 generations. 21 predators are used with weighted sum of objectives as their evaluation measures. Although the convergence is good, a proper distribution is missing.

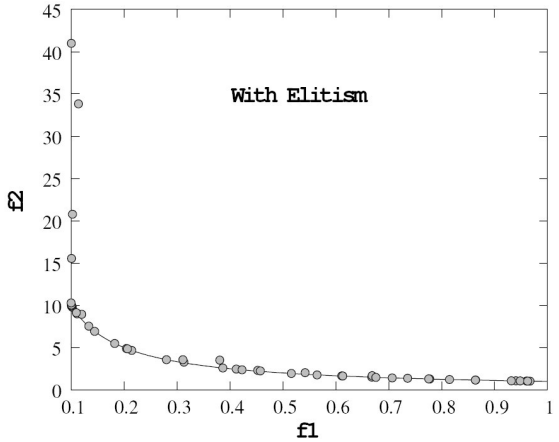

Figure 6: Elite-preservation is introduced.

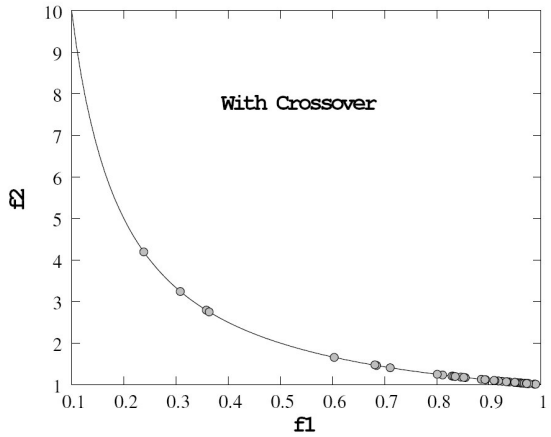

Figure 7: Crossover operation among preys is introduced.

If the newly created offspring is not found to be fit compared to the worst prey, we recreate a new offspring and again compare it with the worst prey. This process is continued at most 10 times. We do not show the results of this study, but this process is found to increase the convergence rate of the proposed algorithm. However, this extension is added in the procedure discussed in the next subsection.

3.3 Ensuring Adequate Distribution:

In this final modification, each prey is assumed to have an influencing region which is defined by a hyper-cube around it on the objective space. The offspring is not accepted if it is created within the influencing region of any existing prey. This is similar to the epsilon-dominance concept used elsewhere [15]. This explicit diversity-preserving procedure forces a good distribution of preys to exist on the Pareto-optimal front. Although the procedure demands the user to suggest the window sizes for the influencing box, in practice they are desired and can be set according to the required precision in the objective values. Figure 8 shows the final population with identical parameter setting as before. The window sizes used in this simulation are 0.01 for both first and second objectives. The figure clearly shows the efficacy of the combined proposed procedure.

Both the convergence to the true Pareto-optimal front and the obtained diversity are better than any of the procedures discussed before.

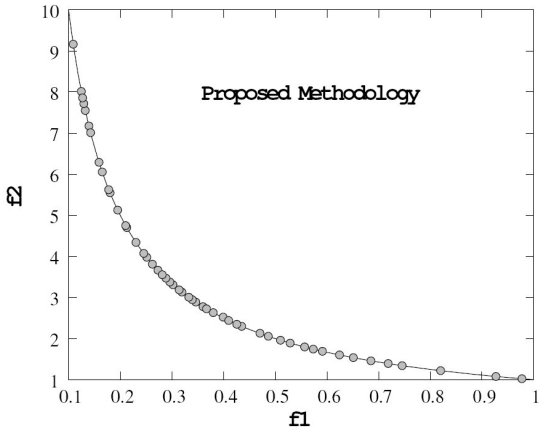

Figure 8: Proposed methodology with combined crossover, elite-preservation and diversity preservation.

3.4 Proposed Predator-Prey Algorithm:

Thus the combined predator-prey procedure has the following step-by-step procedure:

Step 1: Initialize set of preys randomly between the variable limits.

Step 2: Place these preys on the vertices of undirected connected graph.

Step 3: Place predators randomly on the vertices of the graph.

Step 4: Assign each predator with a distinct weighted sum of objectives uniformly created within [0,1]x[0,1]x…x[0,1], so that the sum of weights is one.

Step 5: Evaluate preys around each predator and select the worst prey.

Step 6: Create two offspring by applying a crossover operation between the first and the second best preys in the neighborhood of the worst prey. Randomly choose one of the two offspring and mutate it to create the child solution.

Step 7: Child acceptance criteria:

> *Step 7a:* If the child solution weakly dominates all existing preys or child solution is non-dominated with all existing preys, child becomes a candidate to replace the worst prey. If the child is not within the influencing region of any existing prey, it replaces the worst prey. Predator also moves to the position of the worst prey.
>
> *Step 7b:* Else if the child solution is dominated by any existing prey or the child is within the influencing region of any existing prey, the child is not accepted and a new child is created by Step 6. The creation of new child and its acceptance test are continued a maximum of 10 iterations, after which the worst prey is retained. In this case, predator takes a random walk to any position in the grid.

Step 8: This completes one generation of the predator-prey algorithm. Repeat Steps 5 to 7 for the next generation.

The salient features of the proposed algorithms are as follows:

1. A weighted-sum of objectives per predator is used as a criterion for deleting the worst prey.
2. A crossover between two good solutions and a subsequent mutation are used to create a child solution.
3. The elite preservation and diversity maintenance are ensured by accepting a newly created child only when it weakly dominates all existing preys and it is not within a predefined region from existing preys.

4. MODIFIED ALGORITHM TESTED ON STANDARD TEST PROBLEMS:

First, we apply the modified procedure to two-objective ZDT test problems [1] and later apply to a three-objective DTLZ problem [16]. In all ZDT problems, we have used 21 predators with weighted-sum of objectives uniformed placed within [0,1] and [1,0]. 100 preys and a window size of 0.01 in each objective are always used. In ZDT1 to ZDT4 problems, we have run the procedure till 1,000 generations but for ZDT6 problem we run it till 2,500 generations. Figures 9 till 13 show the final population on the 10-dimensional ZDT problems.

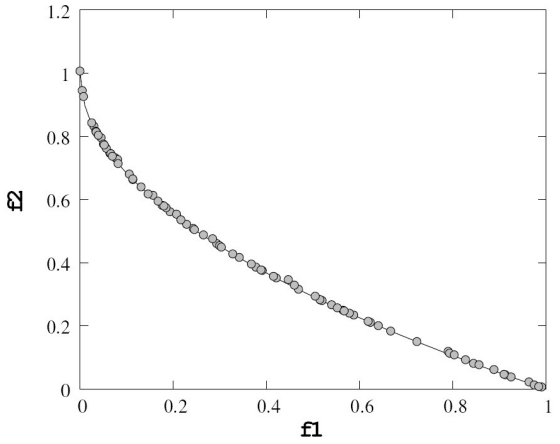

Figure 9: Proposed methodology on 10-variable ZDT1.

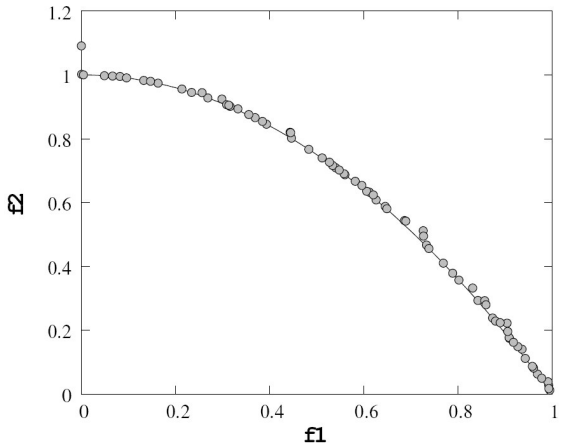

Figure 10: Proposed methodology on 10-variable ZDT2.

The proposed methodology finds almost a uniformly distributed set of solutions for ZDT1 and ZDT2. The convexity of the Pareto-optimal front does not seem to matter to the proposed procedure. Similarly, the disconnectedness of Pareto-optimal frontiers in ZDT3 does not also cause any difficulty to the proposed procedure. However, as can be seen from Figure 12, the approach is not able to overcome all the locally Pareto-optimal fronts present in ZDT4. The procedure gets stuck to one of the locally-optimal front. ZDT6 introduces non-uniformity in solutions along the frontier with more dense solutions towards larger f_1 values. Figure 13 shows that the proposed procedure finds a non-uniform distribution of solutions on the Pareto-optimal front.

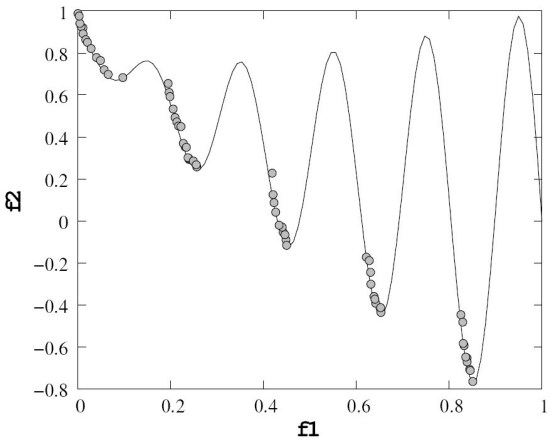

Figure 11: Proposed methodology on 10-variable ZDT3.

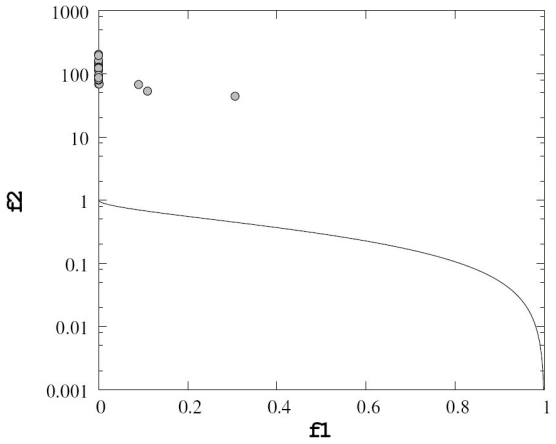

Figure 12: Proposed methodology on 10-variable ZDT4.

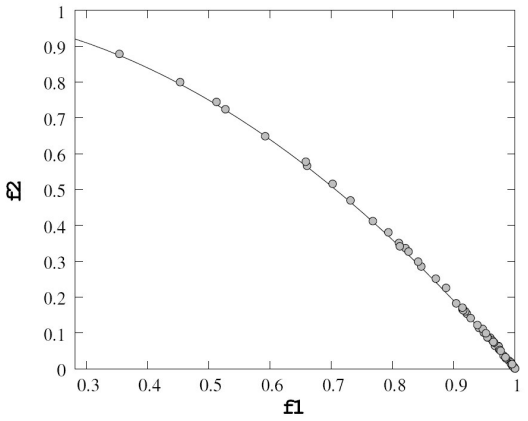

Figure 13: Proposed methodology on 10-variable ZDT6.

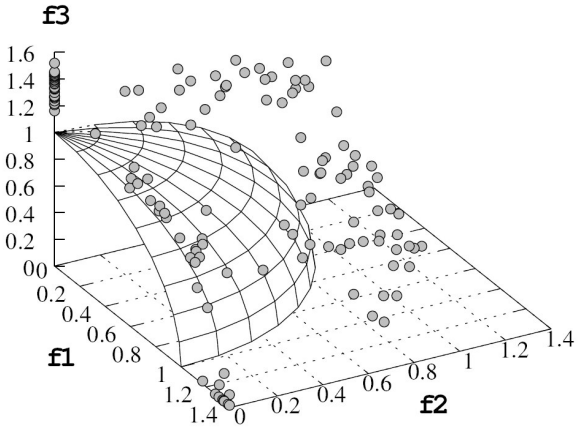

Figure 14: Proposed methodology on 12-variable DTLZ2.

Figure 14 shows the final population on DTLZ2 with 12 variables. Here, 150 preys and 15 predators are used and the procedure is run for 1,000 generations. It can be observed that the population after 1,000 generations did not quite reach the true Pareto-optimal front. It is also observed that with an increase in generations, the convergence towards the Pareto-optimal front is slow. This aspect needs further investigation. However, the maintenance of a good distribution of solutions is a hallmark of the proposed procedure.

5. FINDING PREFERRED OPTIMAL SOLUTIONS:

The predator-prey procedure may be better utilized for a different and more pragmatic task in multi-objective problem solving. Instead of finding the complete Pareto-optimal set, a decision-maker may be interested in finding a solution which is on the Pareto-optimal and is closer to a preferred reference objective vector. In such cases, we can use the proposed predator-prey methodology in a unique manner. Let us consider a general case in which preferred Pareto-optimal solutions are to be found for a set of reference

points. The user supplies these reference points in the objective space. For this task, we modify the proposed predator-prey approach in the following manner:

1. Each predator is assigned to one of the preferred points. Multiple predator assignment to a single reference point is also allowed and is recommended.
2. All neighboring preys are divided into two classes: (i) one which dominates the predator and (ii) the remaining prey solutions.
3. If the second set is empty, we declare the prey having the smallest Euclidean distance as the worst prey. Otherwise, we find the prey in the second set having largest Euclidean distance and declare it as the worst prey.
4. The creation of offspring is identical to the proposed methodology. However, if only the created offspring is within a critical distance from any reference point, this offspring can be considered as a candidate for inclusion in the grid. If the offspring is not within the critical distance of any reference point, it is simply discarded.

With these modifications, we apply the procedure to a number of scenarios on the two-objective five-variable ZDT1 test problem. All these results are taken for 300 generations. The number of preys is chosen in proportion to the number of reference points (25 times the number of reference points). In all cases, 10 predators are considered for each reference point.

Figure 15 to 18 show the final population of preys for different scenarios. In each case, the predators (or reference points) are shown using a red `+'. It is interesting to observe how the proposed methodology is able to find a concentrated set of Pareto-optimal solutions near each of the reference points. It is also interesting to note that the procedure works equally well whether the reference point lies insider the feasible objective space or not.

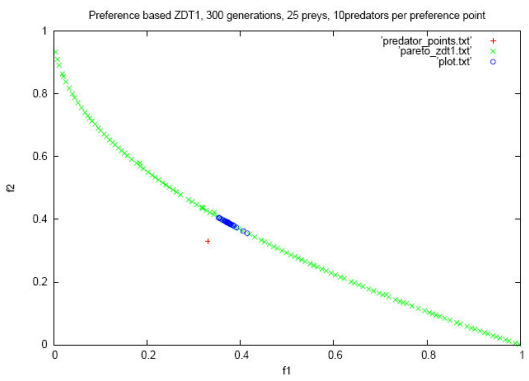

Figure 15: Preferred solutions with an infeasible reference point.

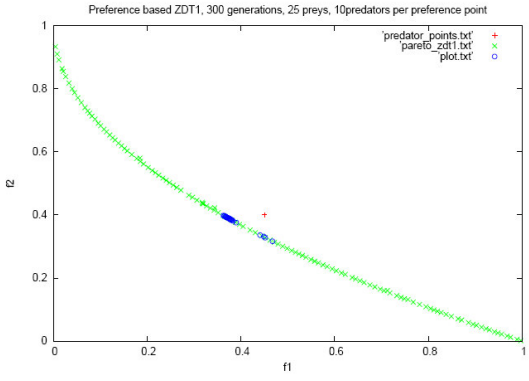

Figure 16: Preferred solutions with a feasible reference point.

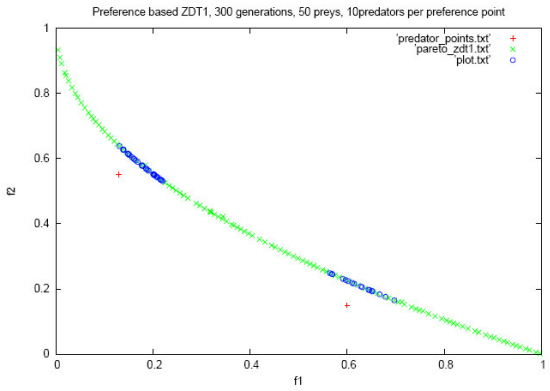

Figure 17: Preferred solutions with two infeasible reference points.

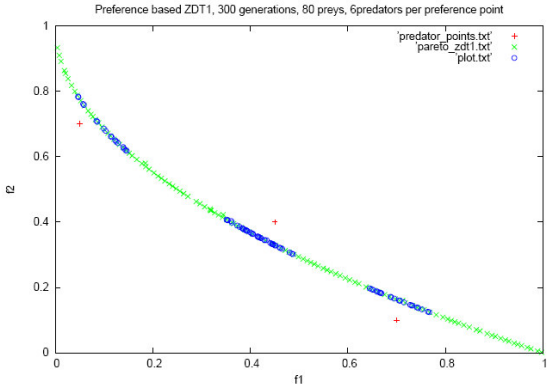

Figure 18: Preferred solutions with two infeasible and one feasible reference points.

6. CONCLUSIONS:

In this paper, we have revisited the predator-prey algorithm originally proposed by a Prof. Hans-Paul Schwefel and his coauthors at the Collaborative Research Laboratory at the University of Dortmund, Germany. The idea of different predators hunting for particular preys based on a different objective function (or goal) was well thought as a candidate multi-objective optimization tool. However, the simplistic implementation was reported to be slow in the past by some researchers. In this paper, we have made a systematic study by first reviewing the past efforts and by suggesting a viable methodology. Although the proposed method may not have a straightforward natural appeal as that in the original model, the proposed procedure is able to find a well-converged and well-distributed set of near Pareto-optimal solutions quickly.

Moreover, the concept of predator and prey interactions has also been utilized well in finding preferred Pareto-optimal solutions near a supplied set of reference points on the objective space. We hope that the methodologies proposed in this paper should motivate other researchers and practitioners to pay more attention to the predator-prey approach of solving multi-objective optimization problems. In the future, the proposed procedure can be extended to solve constrained optimization problems. A hybrid strategy of using the predator-prey approach with another local hill-climbing strategy can also be tried.

REFERENCES:

1. K. Deb. *Multi-objective optimization using evolutionary algorithms.* Chichester, UK: Wiley, 2001.
2. C. A. C. Coello, D. A. VanVeldhuizen, and G. Lamont. *Evolutionary Algorithms for Solving Multi-Objective Problems.* Boston, MA: Kluwer Academic Publishers, 2002.
3. V. Coverstone-Carroll, J. W. Hartmann, and W. J. Mason. Optimal multiobjective low-thurst spacecraft trajectories. *Computer Methods in Applied Mechanics and Engineering,* 186(2-4):387--402, 2000.
4. K. Deb and A. Srinivasan. Innovization: Innovation through optimization. *KanGAL Report Number 2005007*, Kanpur Genetic Algorithms Laboratory, IIT Kanpur, PIN 208016, 2005.

5. K. Deb. Unveiling innovative design principles by means of multiple conflicting objectives. *Engineering Optimization*, 35(5):445-470, 2003.
6. P. D. Surry, N. J. Radcli_e, and I. D. Boyd. A multi-objective approach to constrained optimisation of gas supply networks: The COMOGA method. In Evolutionary Computing. AISB Workshop, pages 166-180. Springer-Verlag, 1995.
7. S. Bleuler, M. Brack, and E. Zitzler. Multiobjective genetic programming: Reducing bloat using spea2. In Proceedings of the 2001 Congress on Evolutionary Computation, pages 536-543, 2001.
8. K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182-197, 2002.
9. E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K. C. Giannakoglou, D. T. Tsahalis, J. Periaux, K. D. Papailiou, and T. Fogarty, editors, Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems, pages 95-100, Athens, Greece, 2001.
10. NSGA-II code in C: http://www.iitk.ac.in/kangal/soft.htm
11. PISA software: http://www.tik.ee.ethz.ch/pisa/
12. M. Laumanns, G. Rudolph, and H. P. Schwefel. A spatial predator-prey approach to multi-objective optimization: A preliminary study. In Proceedings of the Parallel Problem Solving from Nature, V, pages 241-249, 1998.
13. Xiaodong Li, "A real-coded predator-prey genetic algorithm for multiobjective optimization", *EMO-2003*, pp. 207-221, (2003).
14. Pettersson, F., Chakraborti, N. and Saxen, H. "A genetic algorithms based multi-objective neural net applied to noisy blast furnace data", *Applied Soft Computing*, (ASOC-202), (2005).
15. K. Deb, M. Mohan, and S. Mishra. Evaluating the ε-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions. *Evolutionary Computation Journal*, 13(4):501-525, 2005.
16. K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable test problems for evolutionary multi-objective optimization. In A. Abraham, L. Jain, and R. Goldberg, editors, *Evolutionary Multiobjective Optimization*, pages 105-145. London: Springer-Verlag, 2005.