A real-coded predator-prey genetic algorithm for multiobjective optimization

Xiaodong Li

School of Computer Science and Information Technology
RMIT University, GPO Box 2476v
Melbourne, VIC 3001, Australia
xiaodong@cs.rmit.edu.au

Abstract. This paper proposes a real-coded predator-prey GA for multiobjective optimization (RCPPGA). The model takes its inspiration from the spatial predator-prey dynamics observed in nature. RCPPGA differs itself from previous similar work by placing a specific emphasis on introducing a more dynamic spatial structure to the predator-prey population. RCPPGA allows dynamic changes of the prey population size depending on available space and employs a BLX-α crossover operator that encourages a more self-adaptive search. Experiments using two different fitness assignment methods have been carried out, and the results are compared with previous related work. Although RCPPGA does not employ elitism, it has been demonstrated that given a sufficiently large lattice size, RCPPGA can consistently produce and maintain a diverse distribution of nondominated optimal solutions along the Pareto-optimal front even after many generations.

1 Introduction

In recent years, evolutionary algorithms have gained much attention for solving multiobjective optimization problems. In contrast to conventional multicriteria analysis models, evolutionary algorithms are population based, hence they are capable of evolving multiple solutions simultaneously approaching the non-dominated Pareto front in a single run [1]. In order to find a good set of Pareto-optimal solutions, an efficient evolutionary algorithm for multiobjective optimization must achieve two goals - to ensure a good convergence to Pareto-optimal front, and to maintain a set of solutions as diverse as possible along the Pareto front [1]. The benefit of such approach is that it gives the decision-maker the freedom to choose from many alternative solutions.

This second goal of maintaining a diverse set of solutions is unique to multiobjective optimization. As only when we have a diverse set of solutions, we can provide a good set of trade-off solutions for the decision-maker to choose from. To preserve solution diversity, several niching and non-niching techniques have been proposed [2]. For example fitness sharing has been adopted as a niching method in a number of multiobjective optimization EAs [3],[4]. Among the

non-niching techniques, Laumanns et al. proposed a spatial Predator-Prey Evolutionary Strategy (PPES) for multiobjective optimization [5]. Their preliminary study demonstrated that their predator-prey approach is feasible for multiobjective optimization, though there are some serious drawbacks associated with it. Deb also examined Laumanns' model and introduced a modified predator-prey model using different weighted vectors associated with each predator [1]. Deb showed that the modified model produced a better distribution of Pareto optimal solution than the original PPES.

Along the same line of thought, in this paper a real-coded predator-prey genetic algorithm for multiobjective optimization (RCPPGA) is proposed. Although RCPPGA share some of its similarity with PPES as proposed by Laumanns et al. [5], RCPPGA makes special emphasis on the use of the dynamic spatial structure of the predator-prey populations (which is lacking in the original PPES). By doing this, we can introduce to our model the kind of predator-prey dynamics more analogous to nature. The main objective of this research is to investigate RCPPGA's ability to approximate Pareto-optimal front, and more importantly to see if the model can produce well-distributed nondominated solutions along the Pareto front. This is of particular interest to us because the PPES proposed by Laumanns et al. seems to have difficulty in doing so when it applies to a more difficult two-objective optimization function [5], in which case, after many iterations, PPES converges to solutions that are concentrated only partially on the Pareto front.

2 Background and Related Work

Studies in ecology and in particular population dynamics have shown that population has unique spatial characteristics such as density and distribution [12]. Individuals that make up a population affect one another in various ways. For example, interactions among individuals of a population are often confined within an individual's immediate neighbourhood. Individuals interact in space and time not only with its own species, but also with competitors, predators, and the environment. These properties play a critical role in the evolutionary process of the individuals in an evolving population.

Many evolutionary algorithms have been developed explicitly exploring the use of spatial structure in a GA population. It has been found that the use of spatial structure is especially effective in maintaining a better population diversity, which is critical in improving the performance of many evolutionary algorithms on difficult optimization problems [6], [7]. Cantu-Paz has provided a very good survey in this area [7]. However, most of these algorithms have been developed for single objective optimization, and they commonly use a static structure, whether it is fine-grained or coarse-grained, that is the spatial structure of the population remains unchanged throughout a GA run. A few exceptions are work done by Kirley et al [8], Kirely [9], and Li and Kirley [10], where the authors examined the effects of introducing dynamic ecological features (e.g., disturbances and varying population density) into a fine-grained parallel GA model. It was

found that such ecologically inspired parallel GA models using a dynamic spatial population structure are comparable or better in performance than those without such features, especially when dealing with difficult multi-modal function optimization tasks. Since multi-modal function optimization shares its similarity with multiobjective optimization in many aspects, such observation led to our current investigation of a recently proposed predator-prey GA [11], more specifically its ability in handling multiobjective optimization problems. For definitions and key concepts used in multiobjective optimization, the readers can refer to [1], [2].

Laumanns et al. proposed a spatial predator-prey evolutionary strategy (PPES) for multiobjective optimization [5], which is inspired by the predator-prey model from ecology. In this model, prey representing potential solutions are mapped onto the vertices of a 2-dimensional lattice. Predators are also introduced to the 2d lattice. Each predator is associated with a particular objective function. Each predator performs random walk on the lattice, and chases and kills the weak prey in its nearest neighbourhood according to its associated objective. A detailed description of PPES can be found in [5]. Laumanns et al. suggested that an adaptive step size for the EA is mandatory in order to obtain a reasonably good distribution of solutions on the Pareto front. However, even with this adaptive step size, in this case a decreasing step size $\sigma_{k+1} = 0.99\sigma_k$, their model produced a rather disappointing result. Only some subsets of the Pareto front were obtained. In fact, in Deb's reproduction of the experiment, as the model was run for more iterations, the nondominated solutions produced by the PPES became less and less diverse, eventually even converging to a single solution on the nondominated front [1].

To overcome this difficulty of maintaining a diverse set of non-dominated solutions, Deb suggested a modified version of PPES [1]. Instead of assigning one objective to each predator, a different weighted vector is associated with each predator. Each predator evaluates its neighbouring prey with respect to the weighted sum of the objectives. For instance, for a two-objective optimization problem, each predator is assigned with a weighted vector (w_1, w_2). If 9 predators are used, then the first predator takes a weight vector (1, 0), the 2nd predator takes (0.875, 0.125), and so on, until the 9th predator takes (0, 1). The least-fit (according to the weighted sum) prey individual is eliminated at each iteration. By using a different weighted vector for each predator, this method allows each predator to emphasize solutions on different part of the Pareto-front. Deb's result on the second function of the original study of Lanmanns et al. gave a much improved performance. Even after many iterations, the diversity of prey solutions was still well maintained in the Pareto-front region.

In this study, a real-coded predator-prey GA (RCPPGA) is developed to handle multiobjective optimization problems. The two above described fitness assignment methods, used by Laumanns et al. [5] and Deb [1] respectively, are adopted in RCPPGA. In the following sections, we will refer Laumanns' method as **method 1**, and Deb's method as **method 2**. By employing these two fitness assignment methods, we can then look at whether RCPPGA would have the

same kind of difficulty in obtaining well distributed optimal solutions, as described by Laumanns et al., and verify whether Deb's weighted vector approach is effective in this model.

3 A Real-Coded Predator-Prey GA for Multiobjective Optimization - RCPPGA

We emphasize the spatial dynamics of predator-prey interaction by using a two-dimensional lattice where the prey and predator populations reside and interact. The 2d lattice has its boundaries wrapped around to the opposite edge, therefore eliminating any boundary conditions. The initial populations of prey and predator individuals are randomly generated and distributed across the lattice. As illustrated in Fig. 1, we often start with a large number of prey and a relatively small number of predators in the initial populations. Each prey presents a possible solution, whereas each predator does not represent a solution but is able to roam around in order to keep the prey in check (ie., its task is to kill the least-fit prey in its vicinity).

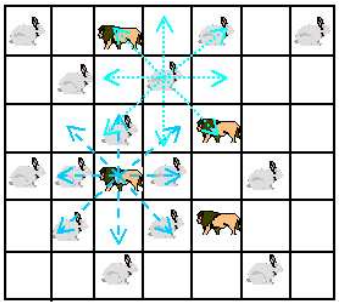


Fig. 1. Predators and prey are randomly distributed across the 2d lattice at the beginning of a run.

After the above initialization, the predator-prey model proceeds in the following steps:

1. Prey and predators are allowed to move from one cell to another on the lattice according to a *randomMoveProbability*, which is normally set to 0.5, so that half the prey would attempt to move one step on the lattice whereas the other half would remain where they were. If the prey were allowed to move, they could choose a random direction, i.e., one of the eight cells in a 8-cell Moore neighbourhood (north, south, ast, west, and plus the four diagonal neighbours), to move into. They then attempt to move. If the cells they are attempting to move into are occupied by other prey or predators, then they try again. Each prey is allowed to try 10 times. If the prey is still unable to find a place to move, it remains where it is.

2. After the prey have moved they are then allowed to breed. Space plays a critical role in this model as each prey can only breed with another prey within its neighbourhood (excluding itself). If the prey has no neighbours it is not allowed to breed. Otherwise the prey is allowed to breed with another randomly selected neighbour to produce an offspring using real crossover and mutation operators (see Section 4.1). The offspring is randomly placed over the entire lattice, which can be seen as migration among different clusters of preys across the lattice. Predators can only kill the least-fit prey in its nearest neighbourhood. 10 attempts are made to place the child on the lattice. If all the attempted cells are occupied, the child is not generated. Note that in this step, the creation of a new child is essentially one function evaluation.
3. The prey population is under constant threat from the predators, which are initially allocated at random across the lattice. Selection pressure is exerted upon the prey population through the predator-prey interaction, that is, predators are given the task of killing the least-fit prey in their vicinity. The predators first look around their neighbourhood to see if there are any prey. If so, the predator selects the least-fit prey and kills it. The predator then moves onto the cell held by that prey. If a predator has no neighbouring prey, it moves in exactly the same way as prey. However it is possible to allow the predators to move more than once per prey step (refer to equation (1)).
4. Go back to step 1), if the number of required evaluations is not reached.

In order to prevent predators from completely wiping out the entire prey population, the following formula is adopted to keep the prey population at an acceptable level:

$$iterations = \lfloor \frac{numPrey_{actual} - numPrey_{preferred}}{num_{predators}} \rfloor \tag{1}$$

where *iterations* is the number of moves the predators may take before the prey can make their moves. A predator can kill at most one prey pre iteration. Basically equation (1) is used to keep the actual number of prey ($numPrey_{actual}$) to a number similar to the preferred number of prey ($numPrey_{preferred}$). The predators are encouraged to minimize the difference between these two values. The floor operator ensures that the predators do not wipe out the prey population entirely. For example, if there are 250 prey, the preferred number of prey is 120, and the number of predators is 20, then the predators would iterate 6 times before the prey have a chance to move and breed again. This is also analogous to the fact that predators are often faster than prey in speed. Another merit of using equation (1) is that as the minimum number of prey (the floor value) is reached, or in another word, the predators become 'slower' in speed, new born prey would have a better chance of survival than otherwise. As a result, the number of prey individuals would start to increase, rather than to continue its decline. This trend would continue until it gets to a point, where the effect of applying equation (1) is once again tipped to be in favour of predators. Using equation (1) in a way provides a mechanism of varying the prey population size dynamically.

One distinct feature of RCPPGA is its explicit implementation of a dynamic spatial structure of the predator-prey populations. In addition to the mating restriction feature as seen in PPES, the predators and prey in RCPPGA can interact via dynamically changing population structure, as predators and prey are capable of moving around on the 2d lattice. Over time, prey clusters of various sizes can be formed naturally by the roaming prey themselves in parallel across the lattice. These different clusters tend to form niches of their own and therefore help preserve prey population diversity over the successive generations. By randomly placing new-born offspring over the entire lattice, we hope to stop potential 'inbreeding' or 'genetic drift' from occurring [12], and meanwhile help continue to maintain a diverse set of solutions until the end of many iterations of a simulation run.

3.1 Selection Mechanism

In RCPPGA, selection pressure is dynamically exerted upon the prey population through the killing and removal of the least-fit prey by the roaming predators. We do not use any direct replacement, for example, fitter offspring directly replacing the less fit ones in the population, as often seen in a typical GA. RCPPGA does not use any elitist mechanism to keep the best-fit prey at each generation. It adopts a rather weak selection method, that is, at each generation, a prey simply breeds with another prey randomly chosen from its neighbourhood.

3.2 Real-Coded Crossover and Mutation Operators

In this real-coded predator-prey GA model, each prey individual represents a chromosome that is a vector of genes, where each gene is a floating point number [13]. For example, a parameter vector corresponding to a GA individual can be represented as $x = (x_1, x_2, \ldots, x_n)$ $(x_i \in [a_i, b_i] \subset \Re, i = 1, \ldots, n)$. The GA works in exactly the same way as the binary counterpart except that the crossover and mutation operations are slightly different.

We follow the suggestion of Wright that a mixed real-coded crossover seems to give better results. This mixed real-coded crossover involves two operations [13].

The first operation uses a real crossover operator, which behaves similarly to a standard crossover. The difference is that instead of swapping binary values, the values in the slots of floating point array (i.e., a chromosome consisting of genes each representing a real-number variable) are swapped. For example, if we have two parents, $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$, and the crossover point is between x_i and x_{i+1}, then one child corresponds to $c_1 = (x_1, x_2, \ldots, x_i, y_{i+1}, \ldots, y_n)$ and the other $c_2 = (y_1, y_2, \ldots, y_i, x_{i+1}, \ldots, x_n)$. We apply this operator to two pareants to produce 50% of gene values of a prey offspring.

The second crossover operator is so called blend crossover operator (BLX-α), first introduced by Eshelman and Schaffer [14]. BLX-α uses two parent solutions x_1^t and x_2^t at generation t, to define a range $[x_1^t - \alpha(x_2^t - x_1^t), x_2^t + \alpha(x_2^t - x_1^t)]$

(assuming $x_1^t < x_2^t$), within which a child solution can be randomly picked. If μ is a random number between 0 and 1, then a child can be generated as follows:

$$x_1^{t+1} = (1-\gamma)x_1^t + \gamma x_2^t \tag{2}$$

where $\gamma = (1+2\alpha)\mu - \alpha$. The above equation can be rewritten as:

$$(x_1^{t+1} - x_1^t) = \gamma(x_2^t - x_1^t). \tag{3}$$

From equation (3) we can observe that if the difference between the two parent solutions x_1^t and x_2^t is small, then the difference between the child x_1^{t+1} and parent x_1^t is also small. This interesting property of BLX-α can be seen as a self-adaptive mechanism for the real-coded GAs [15], because the spread of the population at generation t dictates the spread of the solutions in the population at generation $t+1$. Eshelman and Schaffer reported BLX-0.5 (with α=0.5) gives better results than BLX with other α values. It seems that with α=0.5, BLX provides a nice balance between exploration and exploitation (convergence), therefore we choose to use α=0.5 in this model. We apply BLX-0.5 crossover to the two parents to produce the remaining 50% of the gene values of the prey offspring.

Mutation is applied with a probability to the entire prey population. The mutation operator simply replaces a gene (i.e., a real parameter value) in a chromosome with another floating point number randomly chosen within the bounds of the parameter values.

4 Experiments

4.1 Test Functions

We chose the same two test functions used in Laumanns' PPES [5], and a test function adopted by Deb in his modified PPES [1].

Test function F1:

$$\begin{aligned} &minimize && f_1(x) = x_1^2 + x_2^2, \\ &minimize && f_2(x) = (x_1+2)^2 + x_2^2, \\ &\quad where && -50 \le x_1 \le 50,\ and\ -50 \le x_2 \le 50. \end{aligned}$$

Test function F2:

$$\begin{aligned} &minimize && f_1(x) = -10exp(-0.2\sqrt{x_1^2 + x_2^2}), \\ &minimize && f_2(x) = |\, x_1 \,|^{4/5} + |\, x_2 \,|^{4/5} + 5(sin^3 x_1 + sin^3 x_2), \\ &\quad where && -50 \le x_1 \le 50,\ and\ -50 \le x_2 \le 50. \end{aligned}$$

Test function F3:

$$\begin{aligned} &minimize && f_1(x) = x_1^2, \\ &minimize && f_2(x) = \tfrac{1+x_2^2}{x_1^2}, \\ &\quad where && \sqrt{0.1} \le x_1 \le 1,\ and\ 0 \le x_2 \le \sqrt{5}. \end{aligned}$$

4.2 Experimental Design

In order to compare RCPPGA with the PPES proposed by Laumanns et al and the subsequently modified PPES by Deb [5], [1], experiments were conducted on RCPPGA using the two fitness assignment methods as suggested by them - **method 1**: A different type of predators associated with a different objective [5]; **method 2**: Each predator associated with a different weighted vector [1]. We are interested in finding out whether RCPPGA is able to approximate the Pareto front using the above two fitness assignment methods. In particular, we would like to verify if RCPPGA also has the difficulty in obtaining a good distribution of the nondominated solutions, like the findings on PPES obtained by Laumanns and Deb.

To study the effectiveness of using the dynamic spatial structure of the predator-prey populations, we run RCPPGA in the following experiments using different lattice sizes. A smaller lattice size would restrain the prey population from increasing its size, because the new-born offspring are always competing for the limited vacant space (see equation (1)). On the other hand, if we use the same number of preys in the initial prey population on a larger lattice, the prey population would enjoy a rapid increase in its size, until the lattice is occupied to a threshold governed by equation (1).

RCPPGA is given the following parameter configurations: the lattice size is varied in 3 different sizes, 20x20, 30x30, and 50x50; the number of prey in the initial prey population is 240; the number of predators is 20 and it remains constant throughout a simulation run, however for the fitness assignment method 1, there are two objectives, hence we divide these 20 predators into two types, 10 for each, optimizing according to the two objectives respectively; the $randomMoveProbability$, which is a variable specifying the probability of a prey moving to another cell or remaining stationary, is assigned with a value of 0.5; mutation rate is 0.01; the number of evaluations required is set to 30000 (note that each time a new born is created, it is essentially one function evaluation).

In the following figures, we use a filled circle '•' to indicate the nondominated solutions found at the end of 30000 evaluations, and the plus sign '+' for the dominated solutions. Note that we do not use an external archive to extract the dominated solution in the course of a RCPPGA run. The nondominated solutions shown in the figures are exactly those left in the final prey population when the 30000 evaluations are completed.

5 Results and Analysis

5.1 Effects of Using Different Lattice Sizes

From Fig. 2 - 5, we can see that lattice size has a significant impact on the number of different prey solutions obtained in the final iteration. For a lattice size of 20x20, the RCPPGA converged to only very few nondominated solutions. When the lattice size is increased to 30x30, method 1 gives a much improved performance in terms of the number of different nondominated solutions found.

However, surprisingly, method 2, the weighted vector approach proposed by Deb for the modified PPES, did not perform as well as method 1 for all the 3 test functions. This suggests the inter-dependency of the model parameters, and the fact that parameters have to be chosen carefully in order to obtain satisfactory results.

When the lattice size is increased again to 50x50, both methods 1 and 2 give very good distributions of nondominated solutions for all the 3 test functions. Using a much larger lattice size such as 50x50 in RCPPGA means more prey individuals are allowed to survive. This is because there are more vacant cells available, but the same constant number of predators has to cover a much larger lattice (see Section 3).

One important observation from the above figures is that given a sufficiently large lattice size, RCPPGA seems to be able to obtain a very good distribution of nondominated solutions, even without a direct use of adaptive variation operators such as a decreasing step size suggested in PPES. However RCPPGA does use a BLX-α crossover operator, which has the effect of self-adaptation (see Section 3.2). This crossover operator, along with the use of a migration mechanism (i.e., a random distribution of prey offspring over the entire lattice), seems to be able to maintain a diverse prey population and it does not have the problem of missing out the intermediate solutions as observed in PPES. Implicit niching is carried out via the natural formation of prey clusters of various sizes across the lattice, which allows a diverse set of nondominated solutions obtained and maintained until the last iteration.

Fig. 5 shows that even when a very large lattice of size 80x80 is used, RCPPGA still managed to get a fairly good solution distribution on the Pareto front for F3 within 30000 evaluations, especially for method 1. Note that this time there are many more individuals in the population are not quite converged to the nondominated front, though there are already good approximation and distribution of optimal solutions on the nondominated front. Another interesting observation is that method 1 gives a much more uniformed distribution of both nondominated and dominated solutions in the final iteration than those of method 2. Fig. 5b) shows that method 2 converged heavily towards one end of the Pareto front, but missing out converging to some solutions on the other end. However, as shown in Fig. 5 a), method 1 does not seem to have such a problem.

If we examine closely over Fig. 2 - 5, it can be noted that method 2 seems to be more sensitive to different lattice sizes than method 1. As far as solution distribution is concerned, method 2 performed particularly worse than method 1 on both of a smaller lattice size of 30x30 and a larger lattice size of 80x80 (for F3).

5.2 Dynamic Changes of the Prey Population Size

Fig. 6 shows the dynamic changes of the prey population size when different lattice sizes are used. The larger the lattice size is, the larger the prey population would become. Since the maximum number of evaluations is fixed at 30000, a larger prey population would run for fewer generations than a smaller one. For

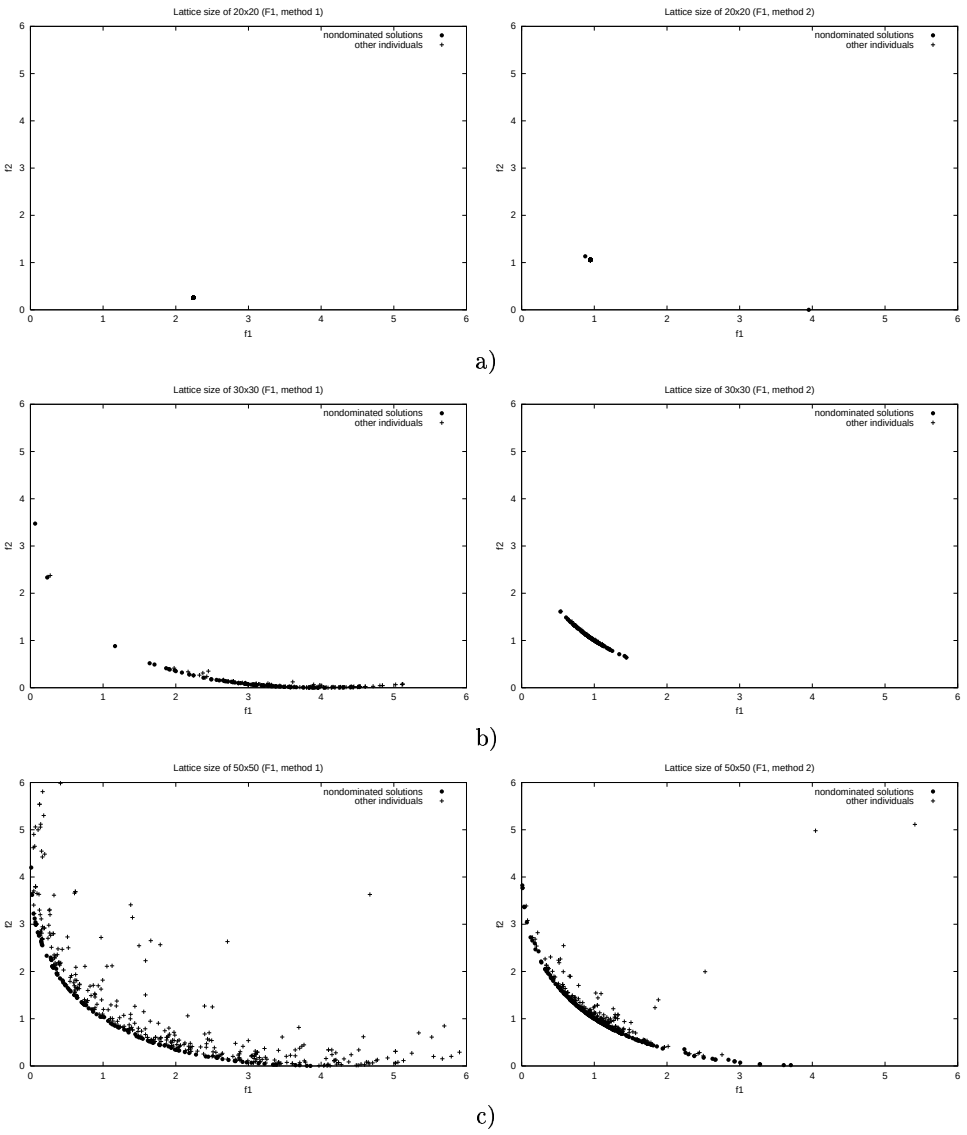


Fig. 2. Prey solutions for F1 using method 1 (left) and method 2 (right) - a) for a lattice of size 20x20, b) 30x30, and c) 50x50.

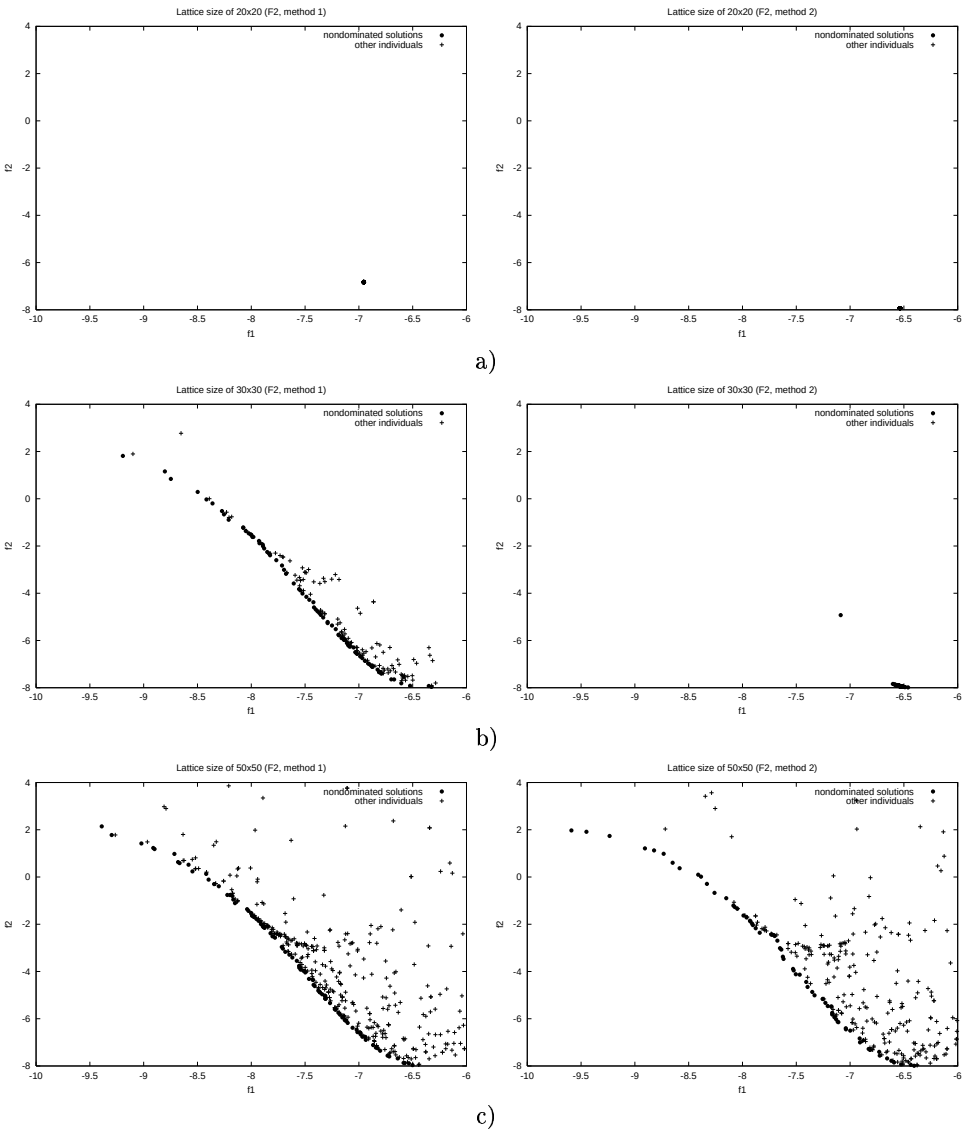


Fig. 3. Prey solutions for F2 using method 1 (left) and method 2 (right) - a) for a lattice of size 20x20, b) 30x30, and c) 50x50.

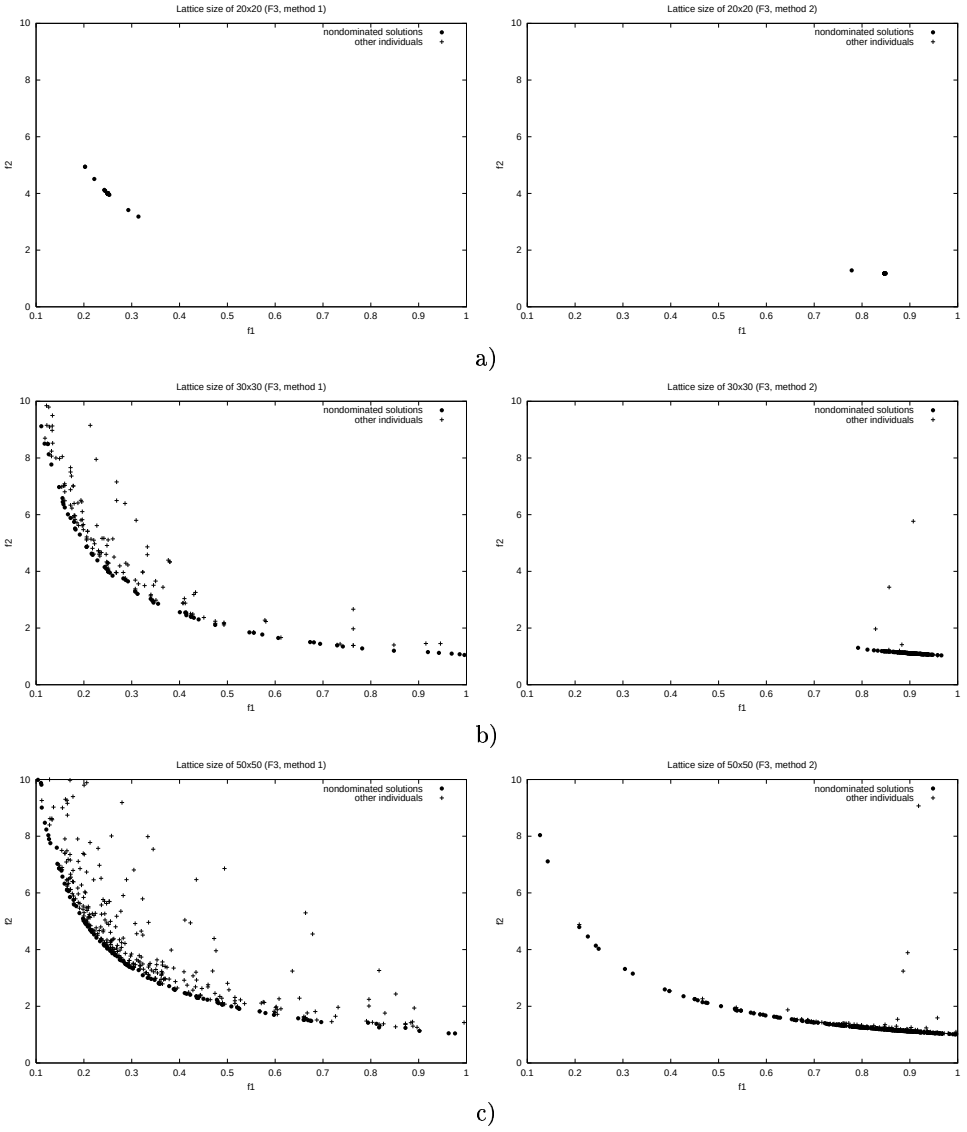
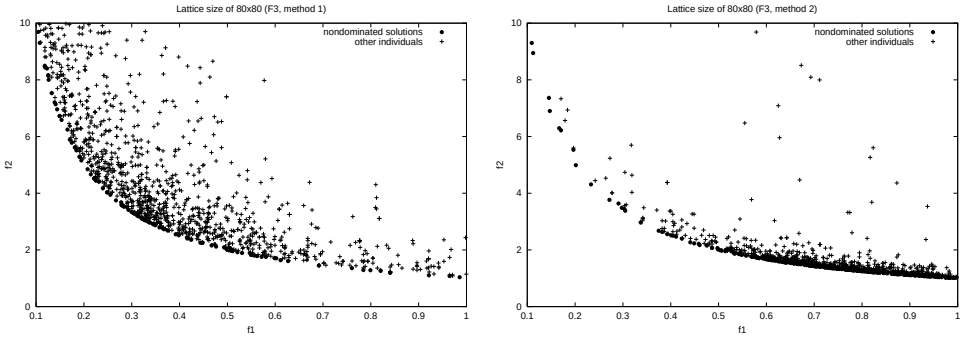
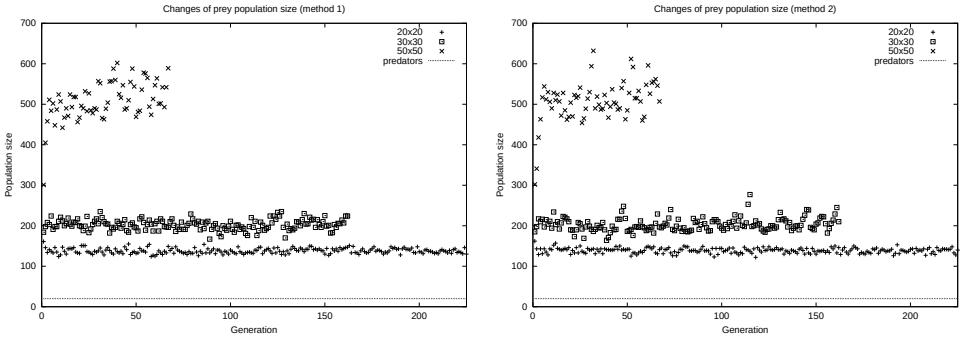


Fig. 4. Prey solutions for F3 using method 1 (left) and method 2 (right) - a) for a lattice of size 20x20, b) 30x30, and c) 50x50.



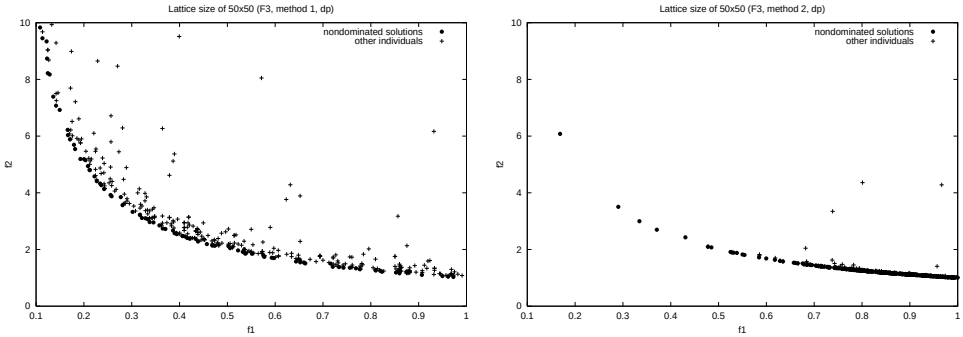
a) b)

Fig. 5. Prey solutions for F3 when a very large lattice size of 80x80 is used - a) method 1, and b) method 2.



a) b)

Fig. 6. The dynamic changes of the prey population size over generations - a) method 1, and b) method 2.



a) b)

Fig. 7. The effects of changing the predator-prey ratio (doubling the number of predators in Fig. 4 c)) - a) method 1, and b) method 2.

example, for a lattice size of 50x50, only 67 generations are run as shown in Figure 6, whereas a lattice of size 30x30 needs 165, and a lattice of size 20x20 needs 225 generations. The lattice of size 50x50 gives the most diverse nondominated solution set however. It can be observed that for the lattice size of 50x50, the number of prey is increased very rapidly at the beginning of a run. Within only a few generations, the prey population has gone from an initial 240 up to around 500, and then fluctuates around this number. It appears that this larger diverse prey population in the early stage of the run provides a basis for RCPPGA to obtain good diverse nondominated solutions towards the end of the run.

5.3 Sensitivity of Predator-Prey Ratio

For method 2, Deb's weighted vector approach, it may appear that by increasing the number of predators, one would hope a better approximation and distribution of the nondominated solutions can be obtained along the Pareto front, however the predator-prey ratio is sensitive to the selection pressure. Fig. 7 shows the effects of changing the predator-prey ratio when we used the same set of parameter values as for Fig. 4 c), but doubled the number of predators. It can be noted that method 2 is more sensitive to the increased number of predators than method 1. The dramatic increased selection pressure occurred when using method 2 resulted in a premature convergence, which in fact further reduced the diversity of the nondominated solutions on the Pareto front. In contrast, method 1 still mananged to obtain well-distributed solutions consistently.

6 Conclusion and Future Work

In this study a real-coded predator-prey GA (RCPPGA) for multiobjective optimization has been proposed. From the experiments carried out over the 3 test functions, it has been shown that RCPPGA is able to produce a good set of diverse nondominated solutions along the Pareto front. The RCPPGA's performance when using two different fitness assignment methods in conjunction with serval lattice sizes have been studied in detail. It has been found that especially with method 1, given a sufficiently large lattice size, RCPPGA can consistently produce and maintain a diverse distribution of nondominated optimal solutions along the Pareto front even after many generations. Method 1 is also empirically shown to be less sensitive to the predator-prey ratio than method 2.

Many current evolutionary algorithms place emphasis on using an elitist mechanism in order to obtain a good optimal solution distribution [1]. By contrast, RCPPGA does not use any elitist mechanism, but still manages to obtain a diverse set of optimal solutions for all the 3 test functions. There is no external archive used in RCPPGA to store nondominated solutions during a run either. The nondominated and dominated solutions are only extracted at the last iteration of a run. One possible explanation for RCPPGA's good performance is that using the self-adaptive BLX-α crossover operator together with the random allocation migration method (see Section 3) is effective and rather non-detrimental as compared with the mutation operators used in the previous studies [5], [1].

In future we will test RCPPGA over more difficult multiobjective optimization problems, especially problems with more than just two objectives. We will also need to include more appropriate statistical measurements on the algorithm performance, for example the Mann-Whitney rank-sum test [16].

References

1. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. John Wiley & Sons, Chichester, UK (2001)
2. Zitzler, E. and Thiele, L.: Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, IEEE Transactions on Evolutionary Computation, 3(4): 257-271, (1999)
3. Hajela, P. and Lin, C.-Y.: Genetic search strategies in multicriterion optimal design, Structural Optimization, vol.4. New York: Springer, (1992) 99-107
4. Horn, J. and Nafpliotis, N.: Multiobjective optimization using niched pareto genetic algorithm, IllGAL Report 93005, Illinois Genetic Algorithm Lab., Univ. Illinois, Urbana-Champaign (1993)
5. Laumanns, M., Rudolph, G. and Schwefel, H. P.: A Spatial Predator-Prey Approach to Multi-Objective Optimization: A Preliminary Study, In: Eiben, A. E., Schoenauer, M. and Schwefel, H.P (eds.): Proceedings of the Parallel Problem Solving From Nature - PPSN V, Springer-Verlag, Amsterdam, Holland, (1998) 241-249
6. Manderick, B. and Spiessens, P.: Fine-grained parallel genetic algorithms. In Proceedings of the Third International Conference on Genetic Algorithms, Morgan Kaufmann (1989) 428-433
7. Cantu-Paz, E.: A Survey of Parallel Genetic Algorithms. Technical Report IlliGAL 97003, University of Illinois at Urbana-Champaign (1997)
8. Kirley, M., Li, X., and Green, D.: Investigation of a cellular genetic algorithm that mimics landscape ecology. In: McKay, R., et al. (eds.): Simulated Evolution and Learning - SEAL98, volume 1585 Lecture Notes in Artificial Intelligence, Springer-Verlag (1998) 90-97
9. Kirley, M. A.: A Cellular Genetic Algorithm with Disturbances: Optimisation Using Dynamic Spatial Interactions. In: Journal of Heuristics. 8(3):321-342 (2002)
10. Li, X. and Kirley, M.: The Effects of Varying Population Density in a Fine-grained Parallel Genetic Algorithm. In: The Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02). Vol: 2, 1709 -1714 (2002).
11. Li, X. and Sutherland, S.: A Cellular Genetic Algorithm Simulating Predator-Prey Interactions. In: Proceeding of the 4th Asia-Pacific Conference on Simulated Evolution And Learning (SEAL'02), Singapore, November 18-22, (2002) (accepted)
12. Smith, R.L. and Smith, T.M.: Elements of Ecology - fourth edition, The Benjamin/Cummings Publishing Company, Inc., Menlo Park, CA 94025 (1998)
13. Wright, A.: Genetic Algorithms for Real Parameter Optimization. In: Rawlin, G.J.E. (eds.): Foundations of Genetic Algorithms 1. Morgan Kaufmann, San Mateo (1991) 205 - 218
14. Eshelman, L. J. and Schaffer, J.: Realcoded genetic algorithms and interval-schemata. In: Foundation of Genetic Algorithms. (1991) 187-202
15. Deb, K. and Beyer, H.: Self-Adaptive Genetic Algorithms with Simulated Binary Crossover, Evolutionary Computation 9(2): 197-221, MIT Press (2001)
16. Knowles, J.D. and Corne, D.W.: Approximating the nondonimated frond using the Pareto Archived Evolution Strategy. Evolutionary Computation, 7(3): 1-26 (2000)