Struct Multidisc Optim (2010) 41:541–554
DOI 10.1007/s00158-009-0433-x

RESEARCH PAPER

Improvements to single-objective constrained predator–prey evolutionary optimization algorithm

Souma Chowdhury · George S. Dulikravich

Received: 26 September 2008 / Revised: 17 August 2009 / Accepted: 22 August 2009 / Published online: 25 September 2009

Abstract In predator–prey algorithm, a relatively small number of predators ("lions") and a much larger number of prey ("antelopes") are randomly placed on a two dimensional lattice with connected ends representing an unfolded surface of a torus. The predators are partially or completely biased towards one or more objectives, based on which each predator kills the weakest prey in its neighborhood. A stronger prey created through evolution replaces this prey. In case of constrained problems, the sum of constraint violations serves as an additional objective. Modifications of the basic predator–prey algorithm have been implemented in this paper regarding the selection procedure, apparent movement of the predators, and mutation strategy. Further modifications have been made making the algorithm capable of handling multiple equality and inequality constraints. The final modified algorithm was tested on standard linear/nonlinear and constrained/unconstrained single-objective optimization problems.

Keywords Predator–prey algorithm · Constrained optimization · Evolutionary algorithms · Crossover · Mutation

This paper has not been published nor submitted for publication anywhere else besides Structural and Multidisciplinary Optimization.

S. Chowdhury · G. S. Dulikravich (✉)
MAIDROC Laboratory, Department of Mechanical and Materials Eng., Florida International University, 10555 West Flagler St., EC 3462, Miami, FL 33174, USA
e-mail: dulikrav@fiu.edu
URL: http://www.eng.fiu.edu/mme/, http://maidroc.fiu.edu, http://www.ISIPSE.net

S. Chowdhury
e-mail: schow003@fiu.edu

1 Introduction

The last few decades have seen the development of optimization algorithms inspired by the principles of natural evolution. These algorithms, often termed Evolutionary Optimization Algorithms (EOAs), use a set of candidate solutions (population space) and follow an iterative procedure to produce a final set of the best compromise solutions, the graphical representation of which is termed as the Pareto front (Deb 2002). In case of single objective problems the Pareto front reduces to a single optimal solution known as the global minimum or global maximum. Genetic algorithm, differential evolution, particle swarm, and predator–prey algorithms are some of the most prominent EOAs.

Hybrid optimization techniques with automatic switching capability among a number of EOAs and classical gradient-based optimization algorithms have also been developed (Dulikravich et al. 1999; Colaço et al. 2005; Moral and Dulikravich 2008) and successfully implemented in multi-disciplinary problems (Martin and Dulikravich 2002).

In 1998, Hans Paul Schwefel proposed a new EOA to search for Pareto-optimal solutions (Laumanns et al. 1998) from a randomly generated initial population of candidate solutions. This algorithm imitates the natural phenomena that a predator kills the weakest prey in its neighborhood, and the next generations of prey that evolve are relatively stronger and more immune to such predator attacks.

In nature, individual predators have different means of tracking their prey, as a result of which their choice of prey might differ. This algorithm mimics such preferential hunting tactics in associating each predator or a group of predators with different objectives. In course of their random movements in the prey neighborhood, each predator

tracks down the weakest local prey, that is, the one which is the most vulnerable to their particular hunting tactics. This refers to the prey which has the worst objective value with respect to that predator. The prey, thus killed, is then replaced by a stronger successor or a child.

Such phenomenon repeating itself over generations leads to evolution of the prey population into stronger species that are more immune to the distinct hunting tactics of the different predators. This, in mathematical terms, reflects improvements of the prey population as a whole with respect to all function objectives. In case of a multi-objective optimization problem, natural selection based on such a method ensures convergence of solutions towards the Pareto front without any direct implementation of a dominance based criterion. This is the major contribution of the predator–prey class of evolutionary algorithms.

In this light, application of the predator–prey selection technique to unconstrained single-objective optimization problems does not furnish any unique benefit. Consequently, in case of single-objective optimization problems, this algorithm acts like a typical genetic algorithm.

However, there is one major difference between the predator–prey algorithm and any other evolutionary algorithm. Absence of any global mixing of the population members results in localized improvements of the prey population in the predator–prey algorithm. This is further facilitated by the evolutionary techniques (crossover and mutation) employed in predator–prey algorithm that establish a localized and adaptive search, thereby enhancing the robustness of this optimization algorithm. This proves to be slower, but a more reliable mechanism of progress towards the global optimum in case of complex single-objective functions, like ones with multiple local optima.

In the predator–prey algorithm, prey, which represent members of the population/solution space are randomly placed (unique integer co-ordinates are randomly generated for each prey) on a two dimensional lattice with connected ends, that is, an unfolded surface of a torus. Predators, which are comparatively fewer in number than prey, are placed at the cell centers of the same 2D lattice. Each predator is completely biased towards one of the objectives, which form the quantitative basis of determining the weakest local prey. After the weakest local prey (the local solution candidate with the lowest value of the fitness function) is identified, it is eliminated (this "prey" is "killed") and a new prey is created through mutation of one of the immediate surviving neighboring prey. While the prey remain stationary, the predators move to a random neighboring location after every generation.

However, this original predator–prey optimization algorithm appears to have difficulty in producing well distributed non-dominated solutions along the Pareto front. Since then, several modifications of the above algorithm have appeared in literature. Deb (2002) suggested an improved version of the algorithm which involved the association of each predator with a weighted sum of objectives instead of one particular objective. Certain new features, namely, the 'elite preservation operator", the 'recombination operator' and the 'diversity preservation operator' were also included. A further modified version of the algorithm was proposed by Li (2003), where a dynamic spatial structure of the predator–prey population was used. It involved the movement of both predators and prey and changing population size of prey. Some other versions of the algorithm have been presented by Grimme and Schmitt (2006) and Silva et al. (2002). The former uses a modified recombination and mutation model. The latter, predominantly a particle swarm optimization algorithm, introduces the concept of predator–prey interactions in the swarm to control the balance between exploration and exploitation, hence improving both diversity and rate of convergence.

Most of the above versions are strictly directed towards unconstrained multiobjective optimization problems. The majority of practical applications of optimization involve constraints. This demands optimization algorithms capable of producing solutions that are both optimum as well as feasible with respect to the problem constraints. There exist very few instances of published applications of any form of the predator–prey algorithm to such real world problems. Nevertheless, since the basic concept of the predator–prey algorithm is significantly different from other standard EOAs, there is sufficient basis to believe that the potentials of this algorithm have not been fully realized.

The fundamental idea of the work presented here is to combine the basic predator–prey algorithm with some advanced features such as the constraint dominance criterion, hypercube sizing and the epidemic operator, to develop a reliable method of solving complex constrained/unconstrained single-objective optimization problems.

There are two benefits of using a multi-objective approach. The algorithm can be used without changing the basic dynamics of the predator–prey interaction and weighted objective association of predators.

In case of solving constrained single-objective problems, the total constraint violation acts as the third objective. The constraint dominance criterion gives preference to selection based on lower constraint violation. On the other hand, the property that the first two objectives are equal to the actual problem objective function leads to two-thirds biasing of predators towards this objective. Both these factors acting together provide a balance between selections of prey (solutions) based on actual objective value as well as its distance from the feasible domain (constraint violation).

This method is somewhat similar to the filter method of constrained optimization, with the dominance criterion biased towards selection based on total constraint

violation. The weighted function association of predators on the other hand creates a counter effect as explained above. However, the version of the algorithm that was found to perform most satisfactorily when dealing with constrained problems involved selection based on 'constraint dominance' criterion instead of the 'weighted sum of objectives' method which makes SOMPP very similar to NSGA II by Deb et al. (2000) with respect to the selection procedure.

This study presents the development of a constrained single-objective version of the modified predator–prey algorithm which involves new features that are expected to promote dependability in terms of convergence of solutions as well as reduction of the number of function evaluations necessary. This single-objective, modified, predator–prey algorithm (SOMPP) has been derived from the basic predator–prey algorithm. Any unconstrained single-objective optimization problem was treated as a two-objective optimization problem, where the second objective is just a clone of the first one. In case of the constrained problems, all the equality and inequality constraints were collaged together to form a third objective and the problem was solved as a three-objective optimization problem (Chowdhury et al. 2009, 2010) where the first two are equivalent and different from the third objective (constraint objective).

2 Single objective modified predator–prey algorithm

Any general constrained single objective test problem is reformulated as follows.

$$\begin{aligned} &\text{Minimize } f_1 = f(X) \\ &\text{Minimize } f_2 = f_1 \end{aligned} \tag{1}$$

subject to

$$\begin{aligned} &g_{ic} \le 0, \quad ic = 1, 2, 3, \ldots, p \\ &h_{ic} = 0, \quad ic = p+1, p+2, \ldots, p+q \\ &p, q \in Nc \end{aligned} \tag{2}$$

Here, X is the vector of design variables, that is,

$$X = (x_1, x_2, .., x_v, \ldots, x_{Nv}) \quad , \quad x_v \in R$$

The inequality and equality constraints are added up to form the third objective

$$\begin{aligned} \text{Minimize } f_3 = &\sum_{ic=1}^{p} \max(g_{ic}, 0) \\ &+ \sum_{ic=p+1}^{p+q} \max((h_{ic} - \varepsilon), 0) \end{aligned} \tag{3}$$

where ε is the tolerance for equality objectives.

In case of unconstrained problems, SOMPP indeed acts as a generic genetic algorithm which selects solutions based on their objective value. However, SOMPP also applies the hypercube technique as a qualification criterion for accepting new/child solutions. This incorporates diversity into the population in the same way as the concept of crowding distance does in case of NSGA II by Deb et al. (2000) and the recently developed Constrained Particle Swarm Optimization by Venter and Haftka (2009).

The initialization and subsequent steps executed by the algorithm in each generation in solving a single-objective optimization problem are sequentially presented below. It should be noted that in case of a maximization problem the function is multiplied by '−1', to convert it into a general minimization problem.

First, a population of N candidate solutions (prey) is created using Sobol's (1976) quasi-random sequence generator to generate their vectors of design variables. Using these values of design variables, objective functions for each candidate solution are evaluated. Sobol's algorithm offers significantly more uniform distribution of random numbers than a typical random number generator routine.

Then, the prey are placed at the nodes of a two dimensional grid with connected ends hence having a toroidal nature. The grid is allowed to adjust its size dynamically according to the population size maintaining the dimensions $I \times J$, where we found after numerical experimentation that the most suitable value for J is $J = 5$. Consequently, I is chosen such that I is the lowest possible integer for which $N < I \times J$. Random members of the prey population are cloned (four or less) if required in order to ensure that all grid points (having integer co-ordinates) are occupied by prey.

Similarly, M predators are placed on the same 2D grid such that they occupy random cell centers (Fig. 1).

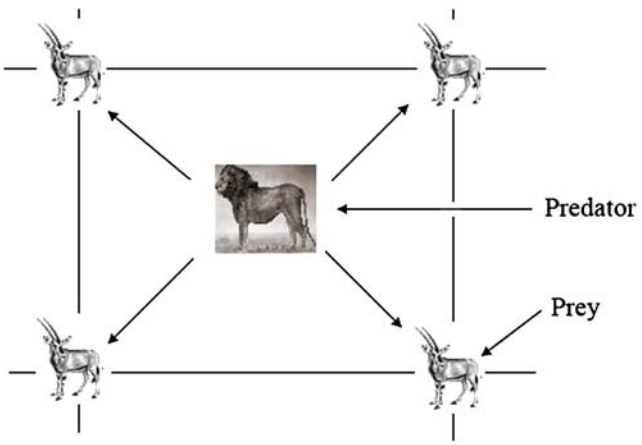

Fig. 1 An active four prey locality/neighborhood on the grid drawn on an unfolded toroidal surface

The value of M is determined by the following empirical formula.

$$M = \max\left(\left(\left\lceil \frac{N}{20} \right\rceil \times N_f\right), 4\right) \tag{4}$$

where, $\lceil r \rceil$ is the lowest integer greater than r, $r \in R^+$, and N_f is the number of objectives. Each predator is associated with a weighted value of the objectives as follows.

$$\begin{aligned} f^j &= \sum_{i=1}^{Nf} w_i^j f_i \\ \sum_{i=1}^{Nf} w_i^j &= 1 \quad \forall j \end{aligned} \tag{5}$$

Here, f^j is the effective objective function value that the jth predator is associated with and w_i^j is respective weight of the ith objective with respect to the jth predator. The weights are distributed uniformly in case of two-objective problems (from (0,1) to (1,0)) and using Sobol's (1976) algorithm in case of problems with more than two objectives (constrained problems).

Thus, the basic SOMPP algorithm was designed to handle problems with more than one objective. The dynamics of the algorithm is also conducive to multi-objective optimization. Hence, an unconstrained single objective optimization problem is treated as a two-objective problem with equal objective values, that is,

$$f_1(X) = f_2(X)$$

This allows one to use the same predator–prey dynamics as in a multi-objective problem. However, mathematically the algorithm will be solving the single objective problem, because each predator will be completely biased towards a single objective (5) since $f^j = f_1 = f_2$ for each predator in the grid.

In case of a constrained single-objective problem, the total constraint violation is treated as the third objective.

This is why the number of objectives is defined by the general notation *Nf, where* $Nf = 2$ for an unconstrained single-objective problem and $Nf = 3$ for a constrained single-objective problem. However, the single-objective MPP (SOMPP) algorithm is significantly different from the multi-objective MPP in some other features, such as the mutation operator, the hypercube operator, rank based predator relocation, nine prey neighborhoods and the epidemic operator. By the virtue of the above features, SOMPP acts as a distinct version of the Modified Predator Prey algorithm suitable for handling single-objective optimization problems.

The presence of three objectives, among which the first two are equivalent and different from the third objective (constraint objective), creates a platform where one can reap substantial benefits from the preferential hunting nature of predators based on their colligation with different objectives. The property that the first two objectives are equal to the actual problem objective function entails a higher probability of prey killings (almost two thirds) based on their weakness with respect to the actual objective. On the other hand, the constraint dominance measure that acts as one of the qualification criterion when accepting new (replacement for the killed) prey, promotes selection based on lower constraint violation.

Notice also that the constraint handling technique used in SOMPP is free of any user-defined constants/coefficients, unlike most penalty function methods (shown in the review by Coello Coello 2002), where the user has to tune different problem specific algorithm coefficients. Consequently, the scope of applying SOMPP universally to any single objective problem with equality and/or inequality constraints seems promising.

Since predators are randomly located at the centers of quadrilateral cells drawn on an unfolded toroidal surface, each neighborhood that contains a predator can be termed as an 'active locality' as shown in Fig. 1. In each of these localities/cells, the value of f as defined by (5) corresponding to the local predator, is calculated for each prey (local fitness of prey). The weakest prey, that is, the prey having the maximum value of f is selected to be killed and replaced by a new prey produced by the crossover of the two strongest neighboring prey and a subsequent mutation of the crossover child.

The blend crossover (BLX-α) (Deb 2002) was used in this case.

$$\begin{aligned} y_v^{(1,t+1)} &= (1-\gamma_v)\, x_v^{(1,t)} + \gamma_v x_v^{(2,t)} \\ \gamma_v &= (1+2\alpha)\, u_v - \alpha \end{aligned} \tag{6}$$

Here, $x_v^{(1,t)}$ and $x_v^{(2,t)}$ are the design variables that define parent solutions, $y_v^{(1,t+1)}$ is the design variable that defines the child solution and u_v is a random number between 0 and 1. A value of 0.5 was used for α as suggested by Deb (2002).

Non-uniform mutation (Deb 2002), as defined below, was used in this algorithm.

$$\begin{aligned} \beta &= 10^{-\left(1+K^{t}/t_{\max}\right)} \\ y_v^{(1,t+1)} &= x_v^{(1,t+1)} \\ &\quad + \tau\left(x_v^{(U)} - x_v^{(L)}\right)\left(1 - r_v^{\left(1-t/t_{\max}\right)^b}\right) \times \beta \end{aligned} \tag{7}$$

Here, 10^{-K} is the terminal order of magnitude of the extent of mutation, $y_v^{(1,t+1)}$ is the child produced by mutation of

the vth variable, $x_v^{(U)}$ and $x_v^{(L)}$ are upper and lower limits of the vth variable, r_v is a random number between 0 and 1, τ takes a Boolean value -1 or 1, each with a probability of 0.5, t and $t_{\max}$ are the number of function evaluations performed until then and maximum allowed number of function evaluations, respectively, while b is the user defined parameter (b = 1.5 determined empirically) and β is the scaling parameter.

The evolutionary operators such as crossover and mutation operator applied in SOMPP are based on the niching strategies used in genetic algorithms. The BLX-α crossover is utilized since it facilitates genetic recombination that is adaptive to the existing diversity in the parent population; a desirable characteristic for Pareto front convergence. Also, non uniform mutation (Michalewicz 1992) is utilized since it provides a uniformly distributed search in the earlier generations and a relatively focused search in the later ones.

The child prey produced by crossover and mutation qualifies to be accepted only if it fulfills the following three criteria:

1. The child is stronger than the worst local prey based on f calculated by (2),
2. The child is non-dominated (Deb 2002) with respect to the other three local prey, and
3. The child is not within the objective space hypercube (Deb 2002) of the remaining three neighboring prey.

Apparently, the treatment of constrained single-objective problems as bi-objective problems with total constraint violation as the second objective is similar to the constrained handling method adopted in other filter algorithms. Nevertheless, the selection criterion is different from the conventional weak dominance criterion used in multi-objective problems. Instead, the constrained dominance criterion as introduced by Deb et al. (2000) is used in SOMPP. The constraint dominance criterion for a minimization problem is defined as follows.

Solution i is said to dominate solution j if:

1. Both solutions are infeasible, and solution i has lower value of constraint violation than solution j (*i.e.*, $f_3^i < f_3^j$)
2. Solution i is feasible and solution j is infeasible.
3. Both solutions are feasible (or problem is unconstrained) and solution i has a lower objective value than solution j (that is, $f_1^i < f_1^j$).

SOMPP, thus, does not follow the actual Pareto approach in searching for optimal solutions. Under such circumstances the predator–prey algorithm demonstrates a desirable balance between selection of solutions based on actual objective value and its distance from the feasible domain. This allows one to incorporate the useful genetic traits of strong infeasible solutions, while driving the prey population towards the feasible domain.

In case of the third criterion, each old local prey is considered to be at the centre of its hypercube, the size of which is dynamically updated with generations and is determined by the following novel equation.

$$\omega = 10^{-\left(2+L\frac{t}{t_{\max}}\right)}$$
$$\eta_i = \omega \times \min\left(f_i^{new\ prey}, f_i^{old\ prey}\right) \tag{8}$$

Here, 10^{-L} is the terminal order of magnitude of relative window size, ω is the window size of the hypercube and η_i is the half side length of the hypercube corresponding to the ith objective. The first two criteria promote convergence towards the global minimum. The third criterion helps in maintaining diversity in the solution space in order to avoid converging to a local minimum. Ten trials were allowed to produce a qualified child that satisfies these three criteria, failing which the worst prey was retained.

Hence, to conclude, selection in SOMPP is chiefly based on constraint dominance. This gives feasibility a preference over optimality, but promotes both simultaneously, which is partially similar to the filter algorithms. At the same time, the mutation operator and the hypercube operator incorporate the traits of niching. Niching has been applied in the field of evolutionary algorithms using various techniques such as dynamic mutation, preselection (Cavicchio 1970), crowding distance concept (Dejong 1975), sharing function model (Goldberg and Richardson 1987), etc. However, SOMPP demonstrates a search radius that is adaptive to the extent of convergence of the population (through adaptive mutation) and a diversity preserving technique (the hypercube operator) adaptive to the current diversity of the population; the simultaneous existence of both is rare in literature. This reinforces SOMPP with the ability to adapt to the complexities of the problem (especially multimodality) at hand.

Thus, we can conclude that the preferential hunting tactics of predators in the predator–prey algorithm do not contribute any unique gain in case of unconstrained single-objective problems. However, when dealing with constrained single-objective problems, such characteristic is highly favorable to ensure simultaneous achievement of feasibility and objective optimization.

Upon completion of the above predator–prey interactions in each active locality, the predators were relocated randomly. A probability based relocation criterion was introduced here, which ensures that each cell is visited, therefore favoring an even distribution of the number of visitations by

a predator to each cell. The predator relocation criterion is defined as follows:

$$\begin{array}{ll} \text{if} & cellcount\,(i,j) > cellcount_{avg} + 1, \quad locate = no \\ \text{else} & , \quad locate = yes \end{array} \tag{9}$$

Here, $cellcount\,(i, j)$ is the number of times predators have visited the cell (i, j) in previous generations, $cellcount_{avg}$ is the average of all $cellcount\,(i, j)$ and (i, j) is the randomly generated location on the 2D lattice. This new feature ensures that every member of the prey population irrespective of its location in the 2D lattice gets fair opportunity of improvement.

At the end of each generation the objective value of the strongest prey (based on dominance criterion) is found and the algorithm checks for termination. The termination criteria are as follows:

1. Maximum allowed number of function evaluations (*fcallmax*) has been exhausted, or
2. The best objective value searched by the algorithm has not changed during the last 100 generations.

The dynamic reduction of the window size of the hypercube and the mean extent of mutation along the course of generations introduces the desirable attribute of 'adaptive shrinkage of the search radii' as solutions converge towards the global optimum.

The above steps summarize the basic version of SOMPP which can be termed as SOMPP Version-1. During the course of further development of SOMPP, other alterations/additional features were also implemented causing minor to significant improvements in its performance. These versions of SOMPP are described in detail as follows.

2.1 SOMPP version-2: rank based predator relocation

Localities with relatively stronger prey were designed to have a higher affinity of attracting predators. The probability '$cellprob_{i,j}$' of locating a predator in a particular locality (co-ordinates i, j generated by a random number generator) is determined as follows.

$$\begin{aligned} cellrank_{i,j} &= \min \begin{pmatrix} rank_{i,j} & rank_{i+1,j} \\ rank_{i+1,j+1} & rank_{i,j+1} \end{pmatrix} \\ cellprob_{i,j} &= \frac{N - cellrank_{i,j}}{N} \end{aligned} \tag{10}$$

Here, $cellrank_{i,j}$ is the rank of the cell/locality (i, j) and $rank_{i,j}$ is the rank of the prey located at the grid point (i, j), ranking being determined on the basis of dominance. N is the total number of prey, hence equal to the maximum rank in the population. This feature introduces substantial amount of elitism into the algorithm thereby speeding up convergence. However, in some cases this might limit the domain of search and hence should be applied carefully.

2.2 SOMPP version-3: nine prey neighbourhood

Instead of the predator being located at the center of a four-vertex quadrilateral cell, the predator is now located on the same grid nodes as prey and allowed to have access to all eight preys around it as well as the prey at that very grid location (Fig. 2). This increases the neighbourhood scope of the predator from four to nine. Since prey are not relocated in SOMPP, this modification facilitates faster communication of genetics among prey irrespective of their location on the unfolded toroidal surface grid, which in turn accelerates the rate of improvement of the prey population as a whole.

2.3 SOMPP version-4: global elitist crossover

Here, the worst prey in each active neighbourhood is replaced by the crossover of the strongest two prey in the entire prey population, instead of the strongest two local prey. Strength of the prey in this case is determined on the basis of the objective value. This significantly decreased the number of function evaluations necessary, but often led to stalling of solutions at the local minima. This might be avoided by selecting the parents for crossover out of the top 'p' percentage of the prey population based on dominance, instead of the two global prey with minimum objective values. Nevertheless, even then the fundamental characteristics

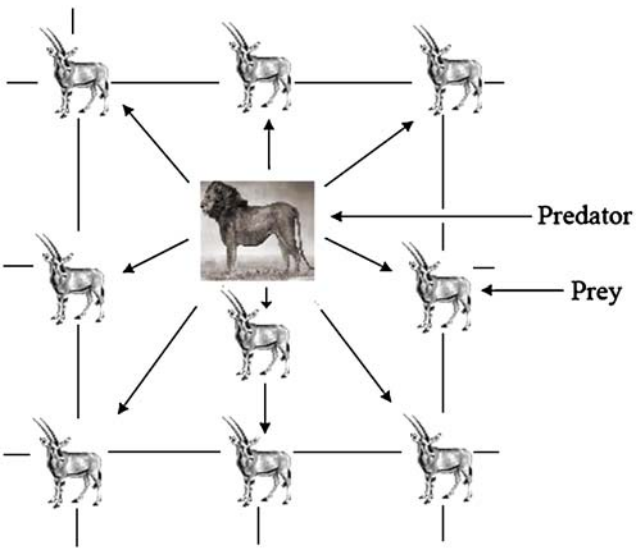

Fig. 2 An active nine prey locality/neighbourhood on the grid drawn on an unfolded toroidal surface

of predator–prey approach, that is localized evolution of solutions, will be lost.

2.4 SOMPP version-5: version-2 and version-3 combined with an epidemical operator

In this version of SOMPP, the concepts of nine-prey active neighborhoods and rank based relocation of predators are implemented simultaneously to promote faster convergence and better communication among the prey. However, the rank for each cell is calculated as the average of the ranks of all the local prey in that cell. In addition to that, to counteract the possibility of convergence to a local minimum, a concept of an epidemic genetic operator was introduced as implemented by Cuco et al. (2008) in the Epidemic Genetic Algorithm. If the objective value of the strongest prey does not change over a certain number of consecutive iterations, a part of the prey population is discarded and replaced with new population generated using Sobol's (1976) quasi-random sequence generator. This is implemented as follows.

If $Nchng > 10$,

1. Rank prey population by dominance.
2. Discard weakest $0.0 < f_w < 1.0$ fraction of the prey population.
3. Set variable limits within the actual specified variable limits as well as suitable to the order of magnitude of the remaining prey design vector values and generate $N \times f_w$ new prey to replace the discarded ones.

Here, *Nchng* is the consecutive number of generations without any change in the objective value of the strongest prey by a relative tolerance of 10e-03. Numerical experiments showed that a high value of f_w ($f_w = 0.9$) should be used for all test cases since whenever the above conditions for the application of the epidemical operator was satisfied, the existing diversity in the population was significantly below that required to produce new solutions. As a result of which, retaining a few representative solutions (stongest of the lot) from the existing population, should be sufficient.

2.5 SOMPP version-6: version-5 with dominance based selection in active neighbourhoods

Here, the relative strength of the prey in an active locality is determined on the basis of the dominance criterion instead of the weighted f value given by (5). In case of unconstrained problems, this has no additional influence because the dominance is merely based on the actual objective value. However, in case of constrained problems, this modification helps significantly in directing solutions into the feasible region first, before the process of minimization takes over. This is because the dominance criterion (Deb 2002) was designed so that feasibility has a preference over minimization. This in turn substantially reduces the domain of search at the later stages making the algorithm more robust and efficient.

The final version of SOMPP (version 6) also incorporates rank based relocation of predators. This is a specific attribute of this single-objective version of predator–prey. Single-objective optimization demands more focused search for optimal solutions compared to multi-objective problems. The rank based relocation ensures that the algorithm does not waste too much time searching sections of the domain which are less likely to contain the optimal solution. However, this can prove to be disadvantageous in cases of highly non-convex or discontinuous functions (like delta functions).

It should be noticed that in SOMPP version 5, the weighted sum of objectives determines the strength of prey, each predator being associated with a different distribution of weights. Whereas in SOMPP version 6 selection is guided by the constraint dominance criterion. This proves to be more favorable for faster convergence of solutions.

3 Numerical experiments

All six versions of SOMPP were implemented using a C++ programming language. The objective functions were evaluated by the corresponding external executable files.

Table 1 Details of three unconstrained single-objective test cases

Problem	N_v	Variable limits	Objective function 1	Analytical solution
Griewank	2	$x_i \in [-600, 600]$	$f(X) = \sum_{i=1}^{m} \frac{x_i^2}{4000} - \prod_{i=1}^{m} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$f(X) = 0,\ x_i = 0$
Rosenbrock	2	$x_i \in [-2.048, 2.048]$	$f(X) = 100\left(x_2 - x_1^2\right)^2 + (1 - x_1)^2$	$f(X) = 0,\ x_i = 1$
Miele-Cantrell	4	$x_i \in [-10, 10]$	$f(X) = \left(e^{(x_2 - x_1)}\right)^4 + 100\,(x_2 - x_1)^6 + \left(\tan^{-1}(x_3 - x_4)\right)^4 + x_1^2$	$f(X) = 0,\ x_1 = 0,\ x_2 = x_3 = x_4 = 1$

N_v number of variables

The final version of the algorithm, that is, SOMPP Version-6 was initially tested on three popular unconstrained single-objective test functions namely Griewank function, Rosenbrock function and Miele-Cantrell function as evaluated by Colaco et al. (2008). Details about these functions are given in Table 1.

The user-defined parameters used in the SOMPP Version-6 algorithm in case of the above three test problems are summarized in Table 2.

The prey population size used here is the 'small set' population size defined by Colaco et al. (2008) as equal to 10 N_v. The crossover probability should be maintained at unity (i.e., 100%) since localized recombination is absolutely necessary for evolution of a population which lacks global mixing of solutions. The mutation probability used is also high (around 0.25, i.e., 25%) which is usual for application of evolutionary algorithms to unconstrained problems. However, specific real world problems might demand a higher or lower mutation probability, which may not be possible to predict a priori without some knowledge of the function topology.

The values of K and L reflect the degree of convergence that the user expects to achieve. However, care should be taken to allow a sufficient number of function evaluations to converge. Otherwise, the local search radii would reduce too much and too soon rendering the algorithm incapable of producing substantially better solutions in subsequent generations. In this case, high values of K and L are used because the above test problems are unconstrained and relatively easy to solve.

The test functions were run until the relative error in the computed minima reduced to 10e-09 or the maximum allowed number of function evaluations was exhausted. The relative error was calculated as follows.

$$relative\ error = \begin{cases} \dfrac{\left|Min_{comp} - Min_{anal}\right|}{Min_{anal}}, & if\ Min_{anal} \neq 0 \\ \left|Min_{comp} - Min_{anal}\right|, & if\ Min_{anal} = 0 \end{cases} \tag{11}$$

Table 2 SOMPP Version-6 user-defined parameters for three single-objective test cases

Parameter	Value
Population size (# prey)	10 N_v
Crossover probability	1.0
Mutation probability	0.25
Maximum allowed function evaluations	10,000
K (mutation)	6
L (hypercube)	10
f_w (epidemic operator)	0.9

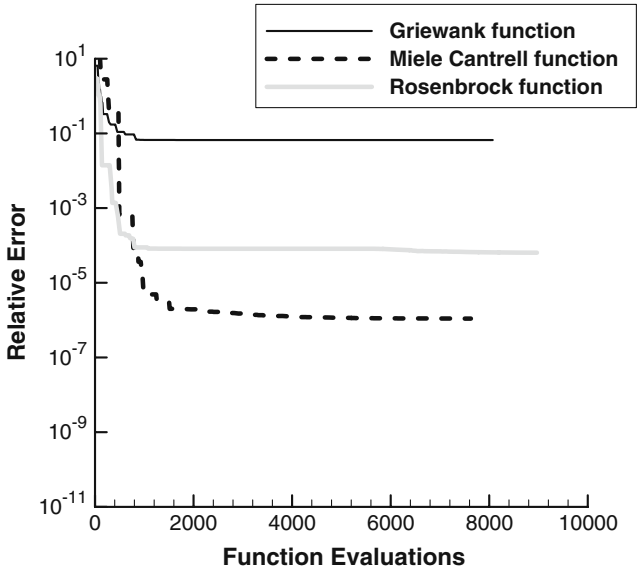

Fig. 3 Convergence history of SOMPP version-1 applied to Griewank, Miele-Cantrell and Rosenbrock functions

The convergence histories of the three test problems are shown in Figs. 3 and 4. It is noticeable that numerous modifications introduced in SOMPP Version-6 made it superior to SOMPP Version-1.

The results for these three test cases for SOMPP Version-1 (Fig. 3) and SOMPP Version-6 (Fig. 4) are shown a priori for ease of latter comparison against results of testing all six version of SOMPP on a much larger set of test functions. Though multiple runs were performed, the outcome of only one of the representative runs is shown here due to obvious constraints in demonstrating convergence histories of multiple runs together on the same graph.

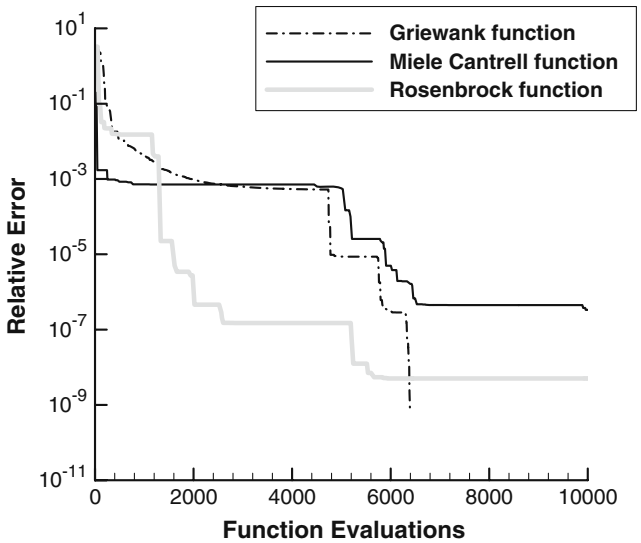

Fig. 4 Convergence history of SOMPP version-6 applied to Griewank, Miele-Cantrell and Rosenbrock functions

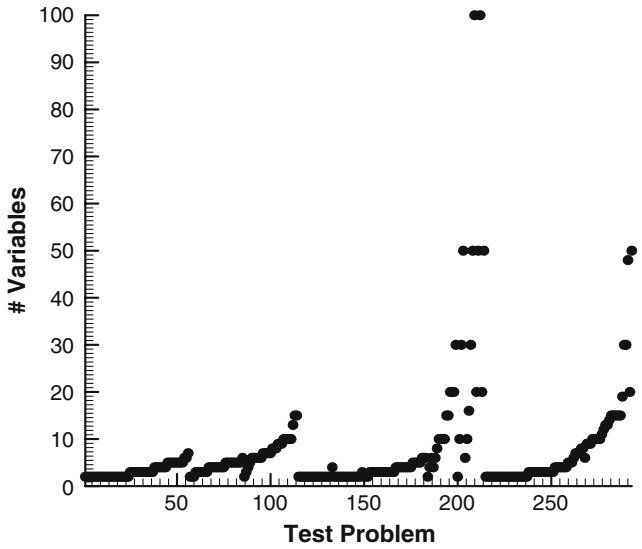

Fig. 5 Number of variables for each of the 293 test cases defined by Hock and Schittkowski (1981)

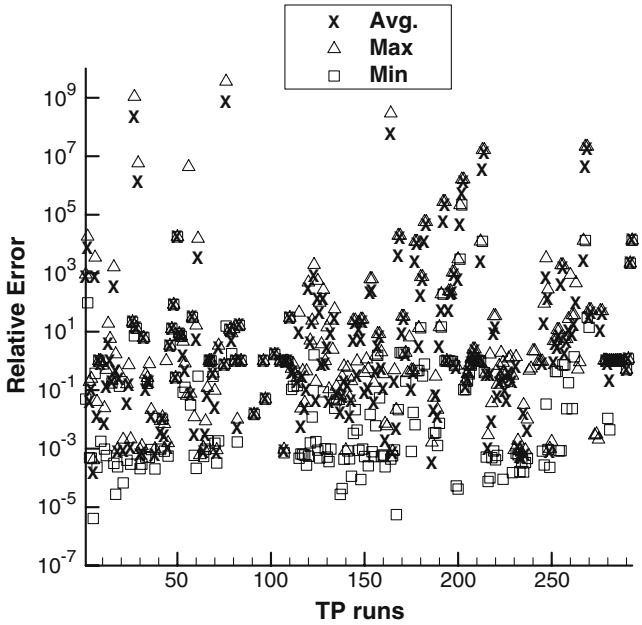

Fig. 6 Relative error of computed minima for the 293 test problems (SOMPP version-1)

Figure 4 shows that the SOMPP Version-6 reduced the relative error by ten orders of magnitude in less than 10,000 function evaluations for the Griewank function. However, in case of the Rosenbrock function and the Miele–Cantrell function the algorithm ran for 10,000 function evaluations to reduce the relative error by ten orders and by only three and a half orders of magnitude, respectively.

Further fine calibration of the extent of mutation and the relative hypercube size together with allowing more function evaluations is likely to achieve better accuracy in finding the global minimum.

In order to test the SOMPP thoroughly, the algorithm in its original version (SOMPP Version-1) was tested on the 293 constrained and unconstrained single objective test cases with known analytic solutions that were derived from the collection of 395 linear/nonlinear test cases (actually 295 test problems) formulated by Hock and Schittkowski (1981) and Schittkowski (1987). The number of variables involved in these cases ranges from 2 to 100 as shown in Fig. 5. The number of inequality and equality constraints range from 0 to 38 and 0 to 6, respectively.

Table 3 SOMPP Version-1 user-defined parameters for the 293 test cases

Parameter	Value
Population size (# prey)	10 N_v
Crossover probability	1.0
Mutation probability	0.1
Maximum allowed function evaluations	20,000
K (mutation)	2
L (hypercube)	4

The user-defined parameters used in the SOMPP Version-1 algorithm in case of the above 293 test problems are summarized in Table 3.

A lower mutation probability is used in this case, because most of the above test cases are constrained and care should be taken to avoid already feasible solutions (near the boundaries of the feasible domain) from leaving the feasible space. Similarly, lower values of K and L were used to impose stricter restrictions on the rate of decrease of search radii,

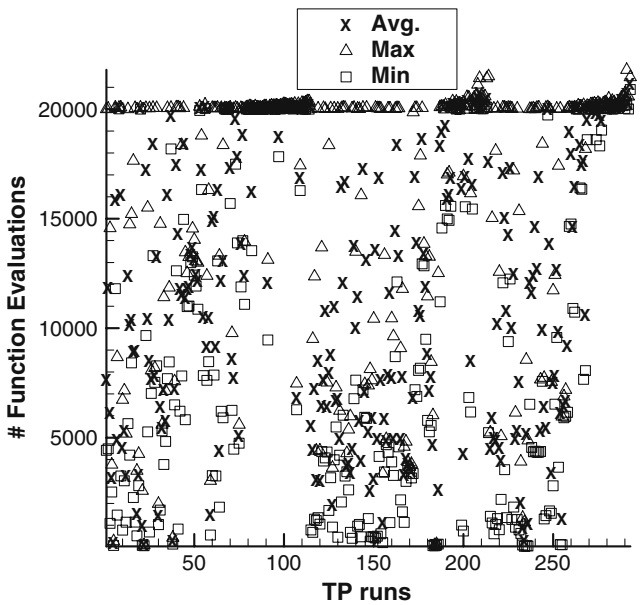

Fig. 7 Number of function evaluations for each of the 293 test problems (SOMPP version-1)

Table 4 Details of the 13 test problems from the set of 293

#	TP	N_v	q	P value
1	1	2	0	0
2	37	3	0	2
3	44	4	0	6
4	55	6	6	0
5	75	4	3	2
6	110	10	0	0
7	112	10	3	0
8	118	15	0	29
9	246	3	0	0
10	251	3	0	1
11	301	50	0	0
12	393	48	2	1
13	395	50	1	0

since a lower degree of convergence is expected for such a set of complex constrained/unconstrained test problems. A relative tolerance of $\varepsilon = 0.001$ was used for equality constraints.

To compensate for performance fluctuations induced by random generators used in creating the initial population and other genetic operators, the algorithm was run five times for each of the 293 test problems resulting in a total of 1,465 test runs. An explicit termination criterion was also implemented when relative error became less than 0.001. The final relative error for the computed minimum and the number of function evaluations exhausted in doing so for each of these test problems can be seen in Figs. 6 and 7. In both of these figures the corresponding maximum, minimum and average (of five runs) are given for each test problem.

It is evident from Fig. 6 that some of the test cases exhibit partial convergence with a relative error of the order of around 1.0. This can be attributed to the presence of either multiple equality or inequality constraints (linear /nonlinear) or both in most of these test problems (Hock and Schittkowski 1981; Schittkowski 1987). Some of the test cases do not converge at all leading to a relative error of orders above unity. This is primarily due to the lack of any

Table 5 SOMPP user-defined parameters for the 13 test cases

Parameter	Value
Population size (# prey)	10 N_v
Crossover probability	1.0
Mutation probability	0.25
Maximum allowed function evaluations	20,000
K (mutation)	3
L (hypercube)	6
f_w (epidemic operator)	0.9

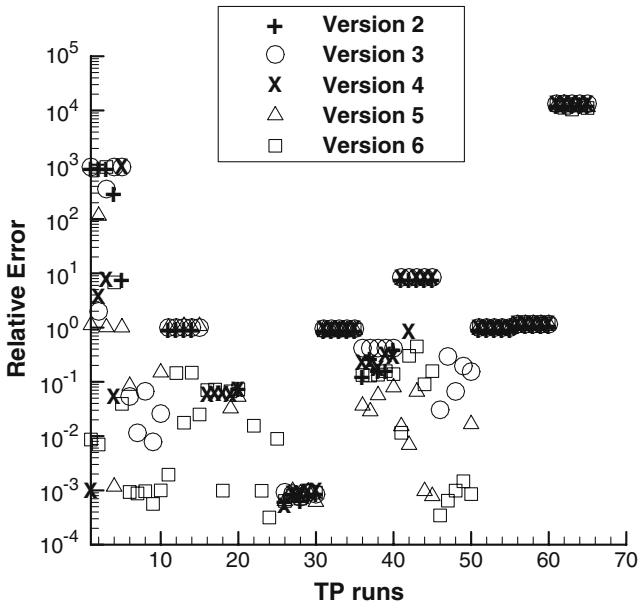

Fig. 8 Relative error of computed minima for the 13 test problems

specified variable limits for some of the variables in the original publications. In such cases, a comprehensive range of −10e10 to +10e10 was assigned for each design variable.

The number of function evaluations varied significantly from problem to problem as seen from Fig. 7. Test problems (TP) from TP-80 onwards till TP-118 (test runs 400–590) have relatively high number of constraints leading to a higher number of function evaluations. Whereas test problems ranging from TP-190 to TP-210 as well as from

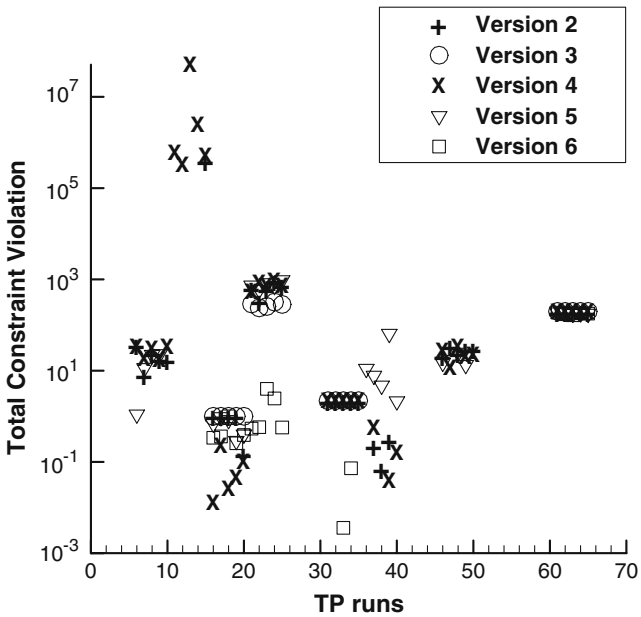

Fig. 9 Total constraint violation for each of the 13 test problems that are constrained

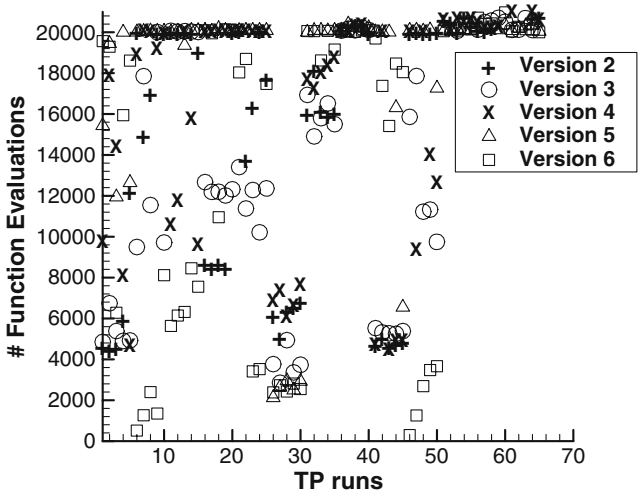

Fig. 10 Number of function evaluations for the 13 test problems

TP-260 to TP-293 have a relatively high number of design variables leading also to a higher number of the objective function evaluations. Though 20000 function evaluations were allowed, some test cases show total executed number of evaluations to be a little more than that. This is because in SOMPP the number of function evaluations in each generation is not limited to the population size (not predictable either) unlike in other evolutionary algorithms. Consequently, the total number of function evaluations might just exceed that allowed in course of the last executed generation. This is also evident from Table 6 presented later in the paper.

Running all 293 test problems in series is computationally extremely time consuming. Consequently, a set of 13 test problems were chosen from among these 293 cases. These 13 test cases involve number of variables ranging from two to 50 (with or without specified limits), number of equality constraints ranging from 0 to 6 and number of inequality constraints ranging from 0 to 38, thereby exhibiting varying degree and nature of complexity. Details pertinent to these test problems are given in Table 4. It should be noted that, compared to Table 3, a higher mutation probability was used to prevent intermediate convergence to local minima and subsequent stagnancy in the region of the local minima. Higher values of K and L were used to achieve better accuracy.

Here, p = number of inequality constraints, q = number of equality constraints.

All the latter five versions of SOMPP (versions 2 to 6) were tested on these 13 test problems. Each of these test problems was run five times on a small population size (10 N_v) as before. The user-defined parameters used in the SOMPP algorithm in case of these 13 test problems are summarized in Table 5.

The relative error of the computed minima, the constraint violation of the computed minima, and the number of function evaluations exhausted for each of the five versions of SOMPP running on each of the 13 test problems thus resulting in 65 runs can be seen in Figs. 8, 9 and 10.

It can be observed from Fig. 8 that SOMPP Version-6 performs better than the other versions of SOMPP in approaching the global minima. It also has the maximum potential in driving solutions into the feasible domain as seen from Fig. 9. In case of some of the constrained problems the data points are not visible in Fig. 9. This is because the constraint violation is zero, that is the final computed minima in these cases are feasible solutions, and hence

Table 6 Output for the 13 test problems with SOMPP Version-6

TP	Computed minima	Actual minima	Relative error	Constraint violation	Number of function evaluations	Computing time (s)
1	0.00701	0.00	0.00701		19,291	989
37	−3,454.06	−3,456	0.00056	0	1,347	69
44	−14.9708	−15.00	0.00195	0	5,635	290
55	6.33959	6.3333	0.00098	0.996963	10,952	568
75	5,176.05	5,174.41	0.00031	2.45536	3,522	182
110	−45.7493	−45.7785	0.00064		2,385	123
112	−0.05151	−0.47761	0.89215	0	20,059	1,045
118	751.617	664.82	0.130556	0	20,031	1,045
246	0.011518	0.00	0.011518		19,696	1,021
251	−3,454.81	−3,456	0.000345	0	294	15
301	0	−50	1.000000		20,052	1,062
393	1.8623	0.86338	1.15699	0	20,712	1,192
395	19,990.6	1.91667	10,428.9	163.789	20,150	1,071

cannot be represented in a logarithmic plot of Fig. 9. The relevant output parameters relating to the most accurate solution (of the five runs for each problem) for SOMPP Version-6 running on the 13 cases are summarized in Table 6.

The significantly low accuracy and inability to find feasible solutions in case of TP-395 can be attributed to the fact that there were no specified variable limits for any of the 50 design variables involved in this problem provided in the original publications (Schittkowski 1987).

SOMPP Version-6 being the most efficient and robust of all the different forms of the SOMPP, was then tested on the entire set of 293 single objective test problems (Hock and Schittkowski 1981; Schittkowski 1987) run five times each. The various user-defined parameters used were the same as given in Table 5. The relative error of the computed minima, the constraint violation of the computed minima and the number of function evaluations exhausted for all the 293 test runs are displayed in Figs. 11, 12 and 13 respectively. In all the three figures, the corresponding maximum, minimum and average (of five runs) are given for each test problem.

It is seen from Fig. 11 that SOMPP Version-6 performs well in achieving relative error of the order of less than 1.0, except for in cases which have a high number of design variables with unspecified variable limits. However, the most prominent improvement of this version of SOMPP is its ability to find the feasible space in case of constrained problems (as shown in Fig.12) irrespective of the number and complexity of the inequality and equality constraints

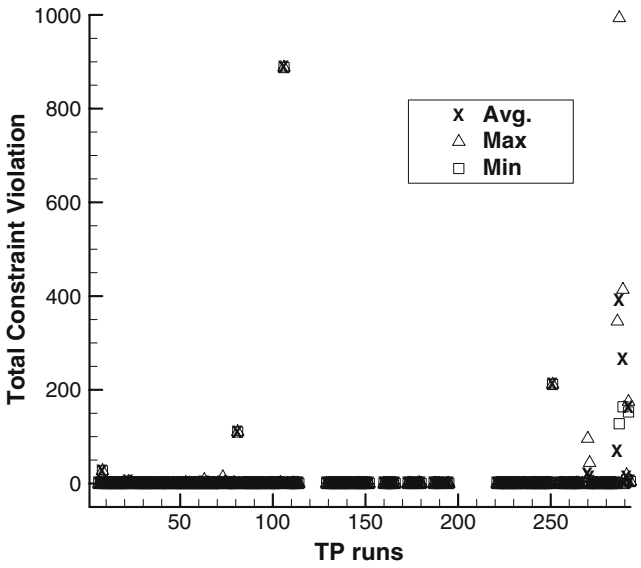

Fig. 12 Total constraint violation for each of the 293 test problems that are constrained (SOMPP version-6)

(whether linear or nonlinear). It should be noted that in many of these constrained problems the initial population is completely in the infeasible space. The inability to converge to the feasible space in case of the last few test problems can be attributed to the involvement of relatively high number of design variables (from 20 to 50) as seen from Fig. 5. The number of function evaluations exhausted by SOMPP Version-6 is relatively high as shown in Fig. 13, which

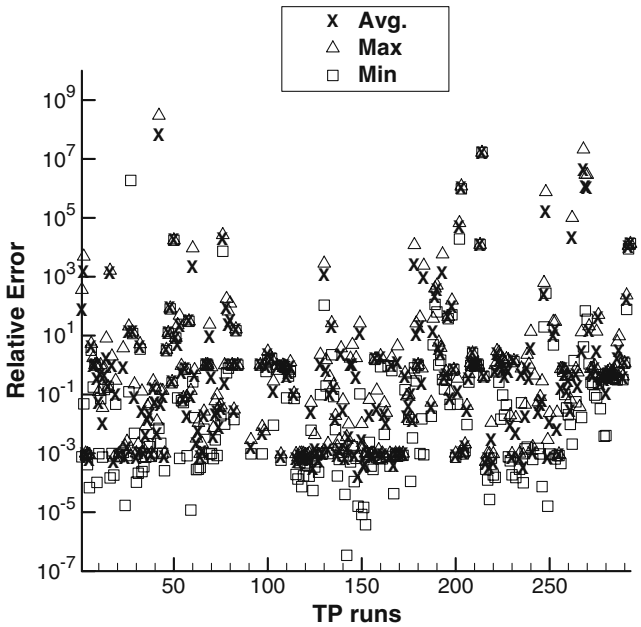

Fig. 11 Relative error of computed minima for the 293 test problems (SOMPP version-6)

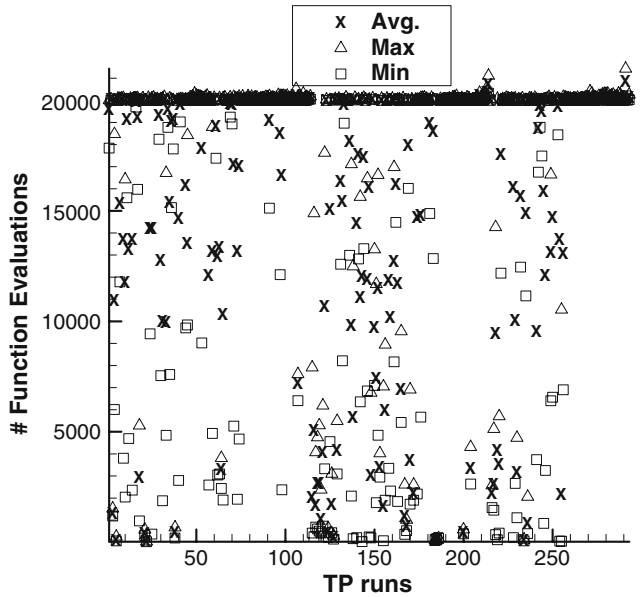

Fig. 13 Number of function evaluations made for the 293 test problems (SOMPP version-6)

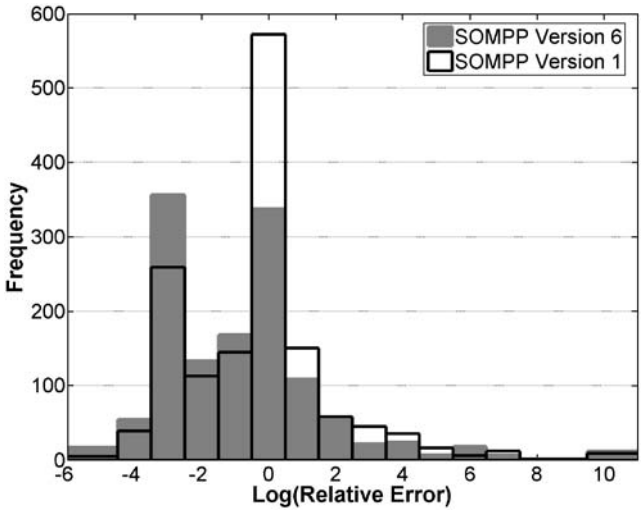

Fig. 14 Comparison of the frequency of occurrence of different orders of magnitude of relative error in the computed minima between SOMPP version-1 and SOMPP version-6. Note: frequency is the number of test runs that converged to that particular order of magnitude of relative error

is expected as a substantial amount of functions evaluations are consumed in successfully searching for the feasible space in case of constrained problems.

The improved performance of SOMPP Version-6 becomes more evident from the histogram presented in Fig. 14.

It is seen from Fig. 14 that in case of SOMPP Version-6, more test cases have converged to relative errors of orders of magnitude less than 1.0 (higher histogram bars for log (relative error) ≤ 0).

4 Conclusion

All versions of the predator–prey algorithm that exist in literature are mostly suited for unconstrained multiobjective optimization problems. Consequently, the predator–prey algorithm in its modified form (SOMPP) is the first of its kind that specifically deals with constrained single-objective optimization problems. It performs well on the popular unconstrained test functions, namely Griewank, Rosenbrock and Miele-Cantrell functions. The 293 single-objective test problems given by Hock and Schittkowski (1981) and Schittkowski (1987) form the most expansive set of single objective test functions (both constrained and unconstrained and linear and nonlinear) available in the literature. SOMPP performs satisfactorily on a large number of these test problems, in driving solutions into the feasible domain and consequently converging to the global minimum, using a relatively frugal population size defined by the 'small set', i.e. ten times the number of design variables Colaco et al. (2008). However, the accuracy of SOMPP is noticeably affected by the absence of specified limits of design variables especially in problems with a large number of design variables.

SOMPP proves expensive in terms of function evaluations when dealing with multiple equality/inequality constraints. This can be attributed to the fact that a substantial amount of function calls are consumed in search of the feasible domain. This expense increases significantly with increase in the dimensionality of the problem, which is however a generic problem with any kind of evolutionary algorithm. Another drawback of SOMPP is that the algorithm demands fine tuning of three user-defined parameters namely the mutation probability, the relative hypercube window size L, and the relative extent of mutation K. Depending upon the problem, a value of 0.05 to 0.25 is suggested for the probability of mutation, whereas values of K and L are subject to the convergence expected with $L - K \geq 2$ always. Nevertheless, coupling SOMPP with an efficient response surface model that interpolates both linear and highly non linear functions in multidimensional spaces (Colaco et al. 2008) is expected to improve the robustness and accuracy of the SOMPP algorithm considerably.

Acknowledgments The authors are grateful for the financial support provided for this work by the US Air Force Office of Scientific Research under grant FA9550-06-1-0170 monitored by Dr. Todd E. Combs, Dr. Fariba Fahroo and Dr. Donald Hearn and by the US Army Research Office/Materials Division under the contract number W911NF-06-1-0328 monitored by Dr. William M. Mullins. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the US Air Force Office of Scientific Research, the US Army Research Office or the US Government. The US Government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation thereon.

References

Cavicchio DJ (1970) Adaptive search using simulated evolution. PhD thesis. Ann Arbor, MI: University of Michigan

Chowdhury S, Dulikravich GS, Moral RJ (2009) Modified predator–prey (MPP) algorithm for constrained multi-objective optimization. In: Burczynski T (ed) EUROGEN 2009, Cracow, Poland

Chowdhury S, Dulikravich GS, Moral RJ (2010) Modified predator–prey algorithm for constrained and unconstrained multi-objective optimization problems. Int J Math Model Numer Optim 1(1):1–24

Coello Coello C (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Methods Appl Mech Eng 191(11–12):1245–1287

Colaço MJ, Dulikravich GS, Orlande HRB, Martin TJ (2005) Hybrid optimization with automatic switching among optimization

algorithms. In: Annicchiarico W, Périaux J, Cerrolaza M, Winter G (eds) A chapter in evolutionary algorithms and intelligent tools in engineering optimization. WIT, CIMNE, Barcelona, pp 92–118

Colaco MJ, Dulikravich GS, Sahoo D (2008) A response surface method-based hybrid optimizer. Inverse Probl Sci Eng 16(6):717–741

Cuco APC, Neto AJS, Velho HFC, de Sousa FLD (2008) Solution of an inverse adsorption problem with an epidemic genetic algorithm and the generalized extremal optimization algorithm. Inverse Probl Sci Eng 16(8):901–914

Deb K (2002) Multi-objective optimization using evolutionary algorithms. Wiley, Chichester

Deb K, Agarwal S, Pratap A, Meyarivan T (2000) A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Proceedings of the parallel problem solving from nature VI conference, pp 849–858

Dejong KA (1975) An analysis of the behavior of a class of genetic adaptive systems. PhD thesis. Ann Arbor, MI: University of Michigan. Dissertation Abstracts International 36(10), 5140B (University Microfilms No. 76–9381)

Dulikravich GS, Martin TJ, Dennis BH, Foster NF (1999) Multidisciplinary hybrid constrained GA optimization. In: Miettinen K, Makela MM, Neittaanmaki P, Periaux J (eds) A chapter in EUROGEN'99- evolutionary algorithms in engineering and computer science: recent advances and industrial applications. Wiley, Jyvaskyla, pp 233–259

Goldberg DE, Richardson J (1987) Genetic algorithms with sharing for multimodal function optimization. In: Proceedings of the first international conference on genetic algorithms and their applications, pp 41–49

Grimme C, Schmitt K (2006) Inside a predator–prey model for multiobjective optimization—a second study. In: Proc. genetic and evolutionary computation conf. GECCO 2006. Seattle WA. ACM, New York, pp 707–714

Hock W, Schittkowski K (1981) Test examples for nonlinear programming codes. Lecture notes in economics and mathematical systems, vol 187. Springer, Berlin

Laumanns M, Rudolph G, Schwefel HP (1998) A spatial predator–prey approach to multi-objective optimization: a preliminary study. Lect Notes Comput Sci 1498:241–249

Li X (2003) A real-coded predator–prey genetic algorithm for multiobjective optimization. Lect Notes Comput Sci 2632:69

Martin TJ, Dulikravich GS (2002) Analysis and multidisciplinary optimization of internal coolant networks in turbine blades. AIAA J Propuls Power 18(4):896–906

Michalewicz J (1992) Genetic algorithms + data structures = evolutionary programs. Berlin, Springer-Verlag

Moral MJ, Dulikravich GS (2008) Multi-objective hybrid evolutionary optimization utilizing automatic algorithm switching. AIAA J 46(3):673–700

Schittkowski K (1987) More test examples for nonlinear programming. Lecture notes in economics and mathematical systems, vol 282. Springer, New York

Silva A, Neves E, Costa E (2002) An empirical comparison of particle swarm and predator–prey optimization. Lect Notes Comput Sci 2464:103–110

Sobol IM (1976) Uniformly distributed sequences with an additional uniform property. USSR Comput Math Math Phys 16:236–242

Venter G, Haftka RT (2009) Constrained particle swarm optimization using a bi-objective formulation. Struct Multidisc Optim. doi:10.1007/s00158-009-0380-6