Defence Science Journal, Vol 51, No 3, July 2001, pp. 285-293

Genetic Algorithm for Optimal Weapon Allocation in Multilayer Defence Scenario

Aparna Malhotra and R.K. Jain

Institute for Systems Studies & Analyses, Delhi - 110 054

ABSTRACT

Several heuristic optimisation techniques have been studied in the past to determine the most effective mix of weapons and their allocation to enemy targets in a multilayer defence scenario. This paper discusses a genetic algorithm approach to arrive at improved solutions with reduced computational time. The most important aspect of the new approach is the mapping of the nonlinear optimisation problem into a discrete 0-1 problem. The results demonstrate that if the problem mapping is correct, even a primitive algorithm can yield high quality results to a complex optimisation problem.

Keywords: Heuristic optimisation technique, genetic algorithm, stochastic algorithm, neural network, weapon deployment

1. INTRODUCTION

There is a complex class of optimisation problems, where traditional deterministic and polynomial time algorithms are not applicable. Recently, a number of techniques have been developed to solve such optimisation problems[1,2]. Some of these novel optimisation techniques include nature-based heuristic search techniques which give a fast but sub-optimal solution to a complex problem. Such heuristic techniques often use biased random search or stochastic algorithms, since exhaustive search methods are not efficient when the problem is nonlinear and has many variables. However, the solutions of stochastic algorithms may not be true global optimum, but near-optimum. Stochastic algorithms include simulated annealing, genetic algorithms, etc. Neural network-based techniques (inspired by the functionality of the brain)[3,4] provide another area of potential research. These nature-based algorithms [5] can be applied to a wide range of problems. These are easy to implement and have the potential of finding near-optimal solution quickly.

Continuous hopfield neural network approach[6-8] is a popular technique for solving NP-complete optimisation problems. In this technique, the problem is formulated in terms of a penalty function, which is a weighted sum of cost functions and the constraint terms. The penalty function is mapped to an energy function, whose various local minima correspond to different valid solutions and the global minima correspond to the best solution. The energy function is governed by several parameters that must be set in order for the network to converge. Improper parameter values can drastically affect the network performance (it may become unstable or converge to infeasible solutions). Further, the parameter values are highly problem-dependent and often need to be adjusted every time the data values change. Also, another drawback of this technique is that if the energy function gets trapped in a local minimum, the

Revised 18 October 2000

algorithm does not provide any means of getting out of it.

Another optimisation technique, simulated annealing[9-11], searches for the optimal solution stochastically by making random changes in the system state, and retaining those changes that result in improvement of the solution. This process is analogous to the statistical mechanics of particles of a substance (either solid or liquid). To alleviate the problem of getting trapped at local minima, simulated annealing occasionally allows 'uphill moves' to solutions of higher costs. Unfortunately the random nature of this search process can result in long convergence times, hence this method is inherently slow.

Genetic algorithm[12-16] is a stochastic search algorithm that models the process of natural selection and genetics. It is an iterative algorithm that maintains a pool of feasible solutions at each iteration. Initially, the pool of solutions is generated randomly.

Subsequently, at each iteration, a new pool of solutions is formed applying the so-called genetic operators (reproduction, crossover, mutation) on the solutions from the previous pool. These operators try to mimic the principles of evolution using a fitness function based on the objective function of the problem. A new pool of solutions may consist of old solutions selected because of their high objective function values or new solutions formed by combining other solutions. This process is repeated until convergence is achieved.

In the typical genetic algorithm, the problem variables are generally integers. This causes adjacency. The problem of optimal weapon mix, deployment and allocation in a multilayer defence scenario[17], (a complex weapon target allocation problem) has been formulated into a 0-1 optimisation function[8]. The formulation of the nonlinear problem into a discrete variable fitness function is a very important part of the solution process since it helps in alleviating the adjacency problem. In this paper, genetic algorithm is applied to this well-defined formulation[8]. The approach is very simple and highly generalised.

2. WEAPON ALLOCATION IN MULTILAYER DEFENCE SCENARIO

The process of effectively allocating resources (weapons in this case) against a perceived enemy threat is known as battle management/command control and communication[7,17] (BM/C^3). Before releasing various types of weapons from the inventory, considerations have to be made regarding their total operating cost, manpower required to operate these, etc. Subsequently, deployment of these weapons involves their placement to protect different strategically important assets, taking into consideration the values of these assets and the area available for weapon operation. The moment some information is available about the possible incoming enemy attacking weapons, defending weapons have to be quickly allocated and launched to neutralise the threat. While allocating defending weapons, the factors like the enemy's possible attack plan, the effectiveness of the defending weapons and other required resources have to be considered. The model considers all these factors to formulate an objective function and also takes care of the constraints[8].

2.1 Mathematical Model

For comparison, consider the well-established formulation[8] of the problem of a multiple layer defence in which two types of attacking weapons (Type 1 and Type 2) are aimed at the three different assets (Asset 1, Asset 2, and Asset 3) as depicted in Fig. 1. These assets are defended by two layers, each containing different types of weapons. Attacking weapons which survive the interception

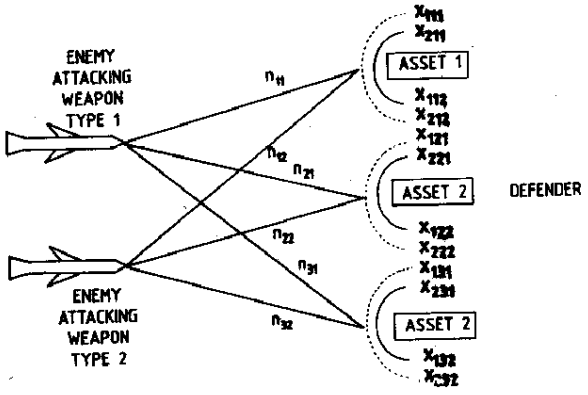

Figure 1. Multiple layer defence

by all layers have a chance to cause damage to the asset. The problem is formulated as follows:

D Types of defending weapons available

S Number of assets

A Types of attacking weapons

k_{dsa} Probability of successful interception by one defending weapon of type d deployed to defend an asset s against an attacking weapon of type a (effectiveness)

x_{dsa} Number of defending weapons of type d deployed to intercept attacking weapon of type a to defend asset s (defence plan)

n_{sa} Number of attacking weapons of type a aimed at asset s (attack plan)

g_{sa} The probability that a single attacking weapon of type a destroys the asset s when it is able to penetrate the defending weapons (damage probability)

v_s Value of asset s

c_d Cost of operating one defending weapon of type d

m_d Manpower required per defending weapon of type d

B_d Number of defending weapons of type d

R_a Number of attacking weapons of type a

G_s Ground area available at asset s

t_d Ground area required by a defending weapon of type d

$C_{\max}$ Maximum operating cost of weapons deployed

$M_{\max d}$ Maximum available manpower to operate defending weapons of type d.

Assuming that the attack plan, effectiveness of defending weapons and the damage probabilities are all known, an objective function corresponding to an optimal defence plan which maximises the total expected surviving value of all the assets, subject to the constraints imposed on the resources of the defender is defined[17]. In such a formulation, the adjacency problem (discussed in Section 2.2) arises[18] because the objective function of x_{dsa} and n_{sa} is integer valued. In this paper instead of integer formulation, a 0-1 formulation[8] of the objective function is used as the fitness function for the genetic algorithm.

2.2 Adjacency Problem

To use a genetic algorithm for solving a problem, all problem variables are generally in the form of binary strings[19] (i.e. a sequence of 1s and 0s). The algorithm consists of application of certain genetic operators (reproduction, crossover and mutation) on these binary strings. Search for better solutions and escape from local minima is highly dependent on these genetic operators and their effectiveness. This often involves flipping the bits with a certain probability. In the above model, suppose the number of defending weapons, x_{dsa} is considered to be an integer variable which is to be determined, and use its binary coding. As x_{dsa} changes from 7 ($7_{10}=[0111]_2$) to 8 ($8_{10}=[1000]_2$), the Hamming distance (number of binary digits that change between successive decimal numbers) will be 4, i.e. all the bits are flipped. Such a significant change using mutation operator is very unlikely even though integer values often have to be changed only slightly to get a better solution. Consequently, mutations in the binary representations of adjacent values of x_{dsa} will not be very effective when trying to find its optimum value incrementally.

Representing the problem variables in gray-codes (which have the characteristics that the Hamming distance between adjacent values is always one)[19] is sometimes helpful but it does not cater for the constraints implicitly. Most of the population strings generated through gray-codes lead to infeasible solutions, thus wasting a lot of computational effort. Hence, there is requirement for a discrete representation of the weapon-target allocation problem, which has some inherent way of by-passing the adjacency problem. An obvious solution is a string representation, where each string would contain x_{dsa} number of 'on' bits and the

remaining 'off' bits. Flipping any one bit of such a string would increase/decrease the value of x_{dsa} by exactly 1.

2.3 Binary Representation of Weapon-Target Allocation Problem

The following parameters which govern the objective function, and consequently determine the size of the strings on which the genetic algorithm can be applied are considered:

(a) Assets s = 1, 2,..., S, where S is the total number of assets.

(b) Defending weapons d_i = 1, 2, ...,D_{max}, where d_i is the i^{th} weapon of type d and D_{max} is the total number of defending weapons of all types, i.e. $D_{max} = B_1 + B_2 + ... + B_D$ where B_k (k = 1, 2, ..., D) is the total number of k^{th} type of defending weapons and D is the total number of defending weapon types.

(c) Attacking weapons a_j = 1, 2,...,A_{max}, where a_j is the j^{th} weapon of type a and A_{max} is the total number of attacking weapon of all types, i.e. $A_{max} = R_1 + R_2 + ... + R_A$, where R_k (k = 1, 2, ..., A) is the total number of k^{th} type of attacking weapon and A is the total number of attacking weapon types.

In order to overcome the adjacency problem in genetic algorithm, the values of variables in the solution space of the weapon-target allocation problem have to be restricted to 0 or 1. The resultant solution space is depicted in form of a matrix in Fig. 2 which is a sequence of D_{max} strings, each of size $A_{max} \times S$, placed one after the other to form a sparse matrix of 1's and 0's.

When the algorithm converges, an element of this matrix, $x_{d_i sa_j}$, which is a bit in a string, will be 'on' if there is an assignment of the i^{th} defending weapon of type d to protect an asset s from the j^{th} attacking weapon of type a. $x_{d_i sa_j}$ will be 'off' if there is no such assignment. The number of 'on' bits in the final strings can then be counted to interpret a solution of the problem. The algorithm works on a population of such matrices and the final output of the algorithm (when it converges) is a similar matrix representing the defence plan. The attack plan (which is known) will also be a 2-D binary string whose element n_{sa_j} being 'on' means that the j^{th} weapon of type a attacks asset s. If there is no such attack, then the corresponding n_{sa_j} is equal to 'off'.

Taking into consideration, k_{dsa} (weapon effectiveness) and g_{sa} (damage probability), the objective function is formulated[8] as follows:

The probability that a single attacking weapon a_j is not intercepted by the defending weapon d_i

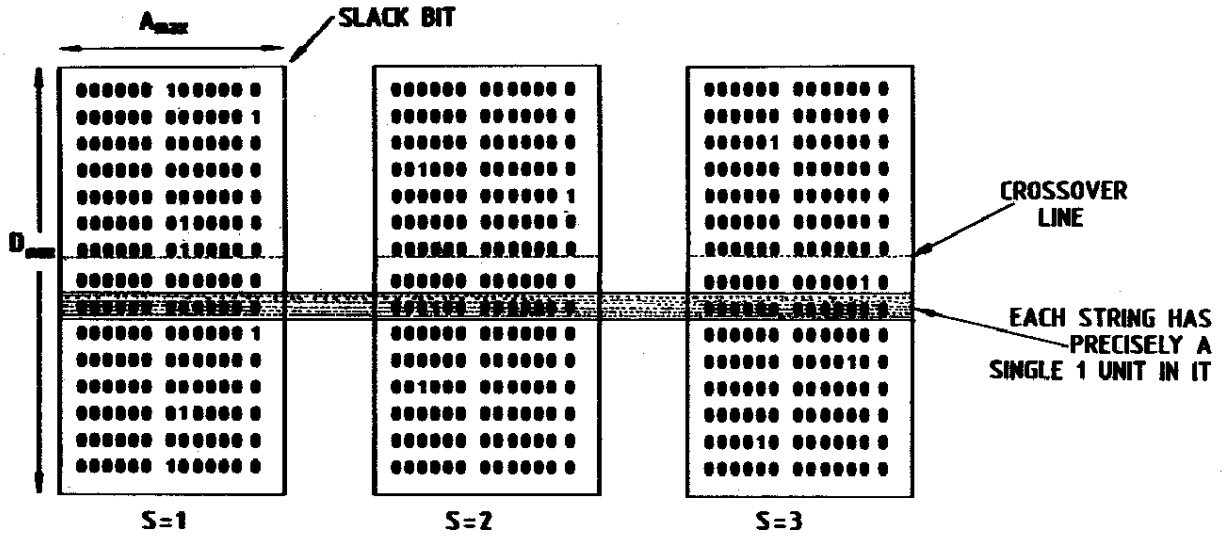

Figure 2. Population matrix corresponding to the optimal defence plan

deployed on asset s, i.e. d_i is not able to defend asset s against a_j is given by:

$$\left[1-k_{dsa}x_{d_i sa_j}\right]$$

The probability that a single attacking weapon a_j is not intercepted by the d_i any defending weapon deployed on asset s is:

$$\prod_{d_i=1}^{D_{\max}}\left[1-k_{dsa}x_{d_i sa_j}\right]$$

Hence the probability that a_j destroys asset s is given by:

$$g_{sa}n_{sa_j}\prod_{d_i=1}^{D_{\max}}\left[1-k_{dsa}x_{d_i sa_j}\right]$$

The survival probability of asset s [Prob(s)], when attacked by all the attacking weapons of all the types is given by:

$$\text{Prob }(s) = \prod_{a_j=1}^{A_{\max}}\left[1-g_{sa}n_{sa_j}\prod_{d_i=1}^{D_{\max}}(1-k_{dsa}x_{d_i sa_j})\right]$$

The total expected surviving value of all the assets, M_{tot}, which is to be maximised is:

$$M_{tot}=\sum_{s=1}^{S}v_s\ \text{Prob}(s)$$

Hence, the objective function (which is the fitness function of the genetic algorithm) to be maximised is given by:

$$\sum_{s=1}^{S}\left\{V_s\prod_{a_j=1}^{A_{\max}}\left[1-g_{sa}n_{sa_j}\prod_{d_i=1}^{D_{\max}}\left(1-k_{dsa}x_{d_i sa_j}\right)\right]\right\} \quad (1)$$

It may be observed that the nonlinear objective function [Eqn (1)] has several independent parameters and the landscape is multimodal, i.e. it has several locally optimal solutions. Obviously, the classical methods of determining the optimal value of a real analytical function of several variables may not be helpful. The multidimensional space of feasible solutions is bounded by the constraint surfaces corresponding to weapon availability, area availability, cost and manpower.

2.4 Inequality Constraints & Slack Bits

If a population matrix contains precisely a single 'on' bit in each string, over-allocation of defending weapons may cause the constraint on resources to be voilated and most of the matrices will represent infeasible allocations. This problem of inequality constraints is catered for by borrowing the concept of slack variables from traditional numerical programming techniques. A slack bit is appended to every sub-string of size $A_{\max}$. In each such string of size $(A_{\max}+1)\times S$, any one bit may be 'on'. The 'on' bit might correspond to the $(A_{\max}\times S)$ main bits or the S slack bits. Hence, atmost $(A_{\max}\times S)$ bits representing weapon assignments can be 'on'. A sequence of $D_{\max}$ number of such strings forms a matrix representing the solution space. The dimension of this matrix can be $[(A_{\max}+1)\times S)]\times D_{\max}$. To cater for the slack bits, the constraints are formulated[8] as follows:

2.4.1 *Weapon Availability Constraint*

Since $D_{\max}=\sum_{d=1}^{D}B_d$, where D is the total type of defending weapons, weapon availability constraint is implicit in the matrix. It was ensured that one defending weapon could be assigned to only one attacking weapon on any one of the assets by having a single bit 'on' in each string. Therefore

$$\sum_{s=1}^{S}\sum_{a_j=1}^{A_{\max}+1}x_{d_i sa_j}=1 \quad (2)$$

2.4.2 *Manpower Constraint*

For a given weapon type, d, this constraint is equivalent to:

$$\sum_{d_i=d_{min}}^{d_{max}} \sum_{s=1}^{S} \sum_{a_j=1}^{A_{max}+1} x_{d_i sa_j} = M_{maxd} / m_d \qquad (3)$$

where d_{min} and d_{max} give the range of d_i for a given weapon type d and

$$(d_{max} - d_{min}) = B_d$$

2.4.3 *Cost Constraint*

$$\sum_{d_i=1}^{D_{max}} \sum_{s=1}^{S} \sum_{a_j=1}^{A_{max}+1} c_d x_{d_i sa_j} = C_{max} \qquad (4)$$

2.4.4 *Area Availability Constraint*

$$\sum_{d_i=1}^{D_{max}} \sum_{a_j=1}^{A_{max}+1} t_d x_{d_i sa_j} = G_s \qquad (5)$$

3. GENETIC ALGORITHM FOR DISCRETE (0-1) PROBLEM

Step 1. Generate initial population

(a) Generate a matrix of dimension $[\{(A_{max} + 1) \times S\} D_{max}]$, such that each row of each matrix has precisely a single 1 at a randomly determined position [may be at the $(A_{max}+1)^{th}$ column].

(b) Check feasibility of this matrix. If infeasible, discard.

(c) Go to *Step* 1(a) and repeat until N*2 feasible matrices are available.

Step 2. From the resultant population, select N best possible solutions.

Step 3. Perform crossover by selecting the crossover points at the end of B_k for k = 1, 2,...,D-1. Take all possible N^D combinations of sub-matrices, e.g. for D = 2, B_1 part of the first matrix and B_2 part of one of the N matrices is linked up to form the resultant matrix. Of these N^D matrices thus generated, consider only N best possible matrices that lead to feasible solutions and discard the remaining.

Step 4. Perform mutations by randomly changing the position of the 'on' bit in each string with mutation probability, P_m.

Step 5. Repeat by going to *Step* 3 until the difference between the maximum fitness of the population between 100 successive iterations is infinitely small.

4. ILLUSTRATION

An example similar to the one defined by Jaiswal[8] is considered to compare the results. Two types of weapons available to defend three assets (Asset 1, Asset 2, and Asset 3) against two types of attacking weapons (Type 1 and Type 2) are considered. It is supposed that the maximum number of defending weapons available of the first type (Type 1) is 7 and that of the second type (Type 2) is 8. The number of attacking weapons of the first and second types are 6 each. The value of the first, second and third assets (Asset 1, Asset 2, and Asset 3) are 400, 300 and 200, respectively. The cost of operating the defending weapons, C_1 and C_2 is 20 and 30, respectively while the maximum cost (C_{max}) is 380. The area required for defending weapons of Type 1(t_1) and Type 2 (t_2) is 34 and 51, respectively.

Table 1. Effectiveness values and damage probabilities

Defending weapon type (d)	Asset (s)	Attacking weapon type (a)	Pd_{dsa}	Pa_{sa}
1	1	1	0.20	0.015
2	1	1	0.60	0.015
1	1	2	0.35	0.055
2	1	2	0.50	0.055
1	2	1	0.25	0.075
2	2	1	0.50	0.075
1	2	2	0.20	0.040
2	2	2	0.45	0.040
1	3	1	0.35	0.060
2	3	1	0.45	0.060
1	3	2	0.25	0.075
2	3	2	0.65	0.075

The area available for Asset 1(G_1), Asset 2(G_2) and Asset 3(G_3) is 225, 150 and 195, respectively. The maximum manpower available for defending weapons 1 and 2 are $M_{max1} = 35$ and $M_{max2} = 32$ while the manpower required for operating each defending weapon weapon Type 1(m_1) and weapon Type 2 (m_2) is 5 and 4, respectively. Effectiveness values of defending weapons and damage probabilities of attacking weapons used for evaluating the fitness function are given in Table 1.

A population matrix consists of 15 strings each containing $(12 + 1) \times 3 = 39$ bits. The attack plan is given in Table 2. An element n_{sa_j} is 1 in the table if the j^{th} weapon of type a attacks asset s. Otherwise the corresponding n_{sa_j} is 0.

Table 2. Attack plan

	Weapon type 1						Weapon type 2					
Asset (s)	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	0	0	0	0	1	1	0	0	0	0
2	0	0	1	1	0	0	0	0	1	1	0	0
3	0	0	0	0	1	1	0	0	0	0	1	1

For this input data, the algorithm converged with the normalised value of the objective function as 0.901756. This is much better as compared to 0.889642 value, which was obtained using continuous Hopfield neural network[8] as applied on the same binary mathematical model. For the attack plan in Table 2, five defending weapons of the first type (Type 1) and seven of the second type (Type 2) are assigned as given in Table 3. The corresponding matrix is represented in Fig. 2. A total of 12 assignments are made. Three slack bits are assigned [2 for first type (Type 1) of defending weapon and 1 for second type (Type 2) of defending weapon].

In this data the manpower constraint does not have any effect on the final value of the cost function since there is always sufficient manpower available. However, if the value of the manpower available to $M_{max1} = 22$ and $M_{max2} = 25$, respectively was reduced, the value obtained was 0.894585. In this case, extra slack bits were assigned to take care of the more constrained manpower requirement. This illustrates how easily any constraint can be imposed and incorporated into the genetic algorithm. Imposing this reduced manpower availability in the neural networks model violates the weapon and area availability constraint. To arrive at a reasonable solution, network parameters have to be re-selected, which is a laborious process of trial and error.

Table 3. Optimal defence plan

Defending weapon type (d)	Asset (s)	Attacking weapon type (a)	Optimal defence plan
1	1	1	0
2	1	1	0
1	1	2	3
2	1	2	2
1	2	1	1
2	2	1	2
1	2	2	0
2	2	2	0
1	3	1	1
2	3	1	1
1	3	2	0
2	3	2	2

5. CONCLUSION

Efforts have been made to evolve an improved methodology which is simple, flexible and rugged for solving a well defined complex weapon-target allocation problem. The methodology so developed is general and can be used for any similar problem. The rate of convergence is very fast provided the mapping of the problem into population strings and fitness function is done correctly.

ACKNOWLEDGEMENTS

The authors are thankful to the referees for their constructive and useful suggestions.

REFERENCES

1. Ansari, N. & Hou, E. Computational intelligence for optimisation. Kluwer Academic Publishers, USA, 1997.

2. Glover, F. & Greenberg, H.J. New approaches for heuristic search: A bilateral linkage with artificial intelligence. *European J. Oper. Res.*, 1989, **39**(2), 119-30.

3. Lippman, R.P. An introduction to computing with neural nets. *IEEE ASSP Mag.*, April 1987, 4-22.

4. Freeman, J.A. & Skapura, D.M. Neural networks, algorithms, applications and programming techniques. Addison Wesley, 1991.

5. Redcliff, N. & Wilson, G. Natural solutions give their best. *New Scientist*, 14th April 1990, 47-50.

6. Tagliarini, G.A.; Fury, J.F. & Page, E.W. Optimisation using neural networks. *IEEE Trans. Comput.*, 1991, **40**(12), 1347-358.

7. Wacholder, E. A neural network-based optimisation algorithm for the static weapon-target assignment problem. *ORSA J. Comput.*, 1989, **4**, 232-45.

8. Jaiswal, N.K. Military operations research, quantitative decision making. Kluwer Academic Publishers, USA, 1997.

9. Fogen, D.B. An introduction to simulated evolutionary optimisation. *IEEE Trans. Neural Netw.*, 1994, **5**(1), 03-14.

10. Johnson, M.E. Simulated annealing and optimisation: Modern algorithms with VLSI, optimal design and missile defence applications. *Am. J. Math. Manage. Sci.*, 1988, **8**(3&4), 205-450.

11. Eglese, R.W. Simulated annealing: A tool for operational research. *European J. Oper. Res.*, 1990, **40**, 271-81.

12. Holland, J.H. Genetic algorithms. *Scientific American*, 1992, 66-72.

13. Srinivas, M. & Patnaik, L.M. Genetic algorithms: A survey. *IEEE Comput. Mag.*, 1994, 17-26.

14. Denning, P.J. Genetic algorithms. *Byte*, 1991, 361-68.

15. Davis, L. Handbook of genetic algorithms. Van Nostrand Reinhold, New York, 1991.

16. Goldberg, D.E. Genetic algorithms in search, optimisation and machine learning, Addison Wesley, New York, 1988.

17. Jaiswal, N.K.; Shrotri, P.K. & Nagabhushana, B.S. Optimal weapon mix, deployment and allocation problems in multiple layer defence. *Am. J. Math. Manage. Sci.*, **13**(1&2), 53-82.

18. Price, K. & Stern, R. Differential evolution. *Dr. Dobb's J.*, April 1997, 18-24.

19. Caruana, R.A. & Schaffer, J.D. Representation and hidden bias: Gray *vs* binary coding for genetic algorithms. *In* Proceedings of the Fifth International Conference on Machine Learning, edited by J. Laird. Morgan Kaufmann, San Mateo, CA, 1998.