Defence Science Journal, Vol. 54, No. 3, July 2004, pp. 395-405


# Hybrid Genetic-simulated Annealing Algorithm for Optimal Weapon Allocation in Multilayer Defence Scenario

Sanjay Bisht

*Institute for Systems Studies & Analyses, Delhi-llQ* 054

**ABSTRACT**

Simulated annealing is one of the several heuristic optimisation techniques, that has been studied in the past to determine the most effective mix of weapons and their allocation to enemy targets in a multilayer defence scenario. Simulated annealing is a general stochastic search algorithm. It is usually employed as an optimisation method to find a near-optimal solution for hard combinatorial optimisation problems, but it is very difficult to give the accuracy of the solution found. To find a better solution, aji often used strategy is to run the algorithm by applying the existing best solution from the population space as the initial starting point. Giving many passes of genetic algorithm can generate the best start-point solution. This paper describes a new hybrid optimisation method, named genetic-simulated annealing, that combines the global crossover operators from genetic algorithm and the local stochastic hill-climbing features from simulated annealing, to arrive at an improved solution with reduced computational time. The basic idea is to use the genetic operators of genetic algorithm to quickly converge the search to a near-global minima/maxima, that will further be refined to a near-optimum solution by simulated anneling using annealing process. The new hybrid algorithm has been applied to optimal weapon allocation in multilayer defence scenario problem to arrive at a better solution than produced by genetic algorithm or simulated annealing alone.

**Keywords:** Heuristic optimisation technique, genetic algorithm, simulated annealing, genetic-simulated annealing, multilayer defence, simulated annealing genetic algorithm, SAGA, GSA, hill-climbing feature

## 1. INTRODUCTION

Most of the combinatorial optimisation problems are proved to be non-deterministic polynomial hard or non-deterministic polynomial complete problems. Genetic algorithm[1,4] and simulated annealing[5,7] provide heuristic algorithm for combinatorial optimisation problems. These have been successfully used for solving nonlinear combinatorial problems.

Genetic algorithm is a stochastic search algorithm, which uses a concept of evolution and natural selection as heuristic. It is an iterative algorithm that retains a pool of feasibly strong solutions at each genetic pass. Initially, the population space is generated randomly. The iterative passes continue, until population is homogenous or satisfies some fitness criterion.

Simulated annealing is another optimisation technique, that searches for the optimal solution stochastically by making random changes in the initial state of the system, and retaining only those changes that result in the improvement to the solution.

Revised 30 July 2003

The word simulated annealing has been derived from the roughly analogous physical process of heating and then slowly cooling a substance to obtain strong crystalline structures. Here, the global minimal cost function corresponds to the ground state of the substance. The simulated annealing process lowers the temperature slowly until the system freezes and no further changes occur. Simulated annealing occasionally allows uphill jumps to solution of higher cost to avoid getting trapped in local minima. Random nature of this search process can result in longer convergence time, and hence, the method is inherently slow. Also, it has no parallelism, whereas genetic annealing has inherent parallelism to arrive at a near-optimal solution. While due to inherent parallelism, genetic annealling is very powerful for searching larger regions of the solution space roughly and globally using the crossover operator, simulated annealing is very powerful for searching local regions of the solution space exhaustively via stochastic hill climbing. Simulated annealing also has the solution refining capability. Combining the global crossover operator of genetic annealing and the local hill-climbing features of simulated annealing, this study proposes a hybrid optimisation algorithm, named as genetic-simulated annealing.

Genetic-simulated annealing has been applied to optimal weapon allocation in a multilayer defence problem, to demonstrate the advantage of genetic-simulated annealing over simulated annealing.

## 2. GENETIC ALGORITHM & SIMULATED ANNEALING

Genetic algorithm[8,9] is an approach for solving combinatorial optimisation problem. Genetic algorithm applies an evolutionary mechanism to optimisation problems. It starts with a population of initial feasible solution, satisfying given constraints. Each solution has a fitness value, which is a measure of the quality of a solution. At each step, known as generation, genetic algorithm produces a set of candidate solutions, known as child solutions, using two types of genetic operators, named mutation operator and crossover operator. It selects good solutions as survivors to the next generation, according to the fitness values. The mutation operator takes a single parent and modifies it randomly in a localised manner, so that it makes a small jump in the solution space. On the other hand, the crossover operator takes two solutions as parents and creates two-child solutions by combining the partial parent solution of the parents. Crossover operator tends to create child solutions, which differs from both the parent solutions. It results in larger jumps in the solution space. Taking a large jump allows genetic algorithm to globally search larger region of the solution space. But genetic algorithm has no explicit ways to produce a sequence of small jumps. Mutation operator creates a single small move, one at a time, instead of a sequence of small moves. As a result, genetic algorithm cannot search local region on the solution space exhaustively. The drawback of genetic algorithm is very well tackled by simulated annealing, which has the capability to search local region in the solution space exhaustively due to its hill-climbing feature.

Simulated annealing is the stochastic iterative improvement method for solving combinatorial optimisation problems. Simulated annealing generates a single sequence of solutions and searches for an optimal solution along this search path. Simulated annealing starts with a given initial solution X_0. At each step, simulated annealing generates a candidate solution X_N by changing a small fraction $X of a current solution X_0. Simulated annealing accepts the candidate solution as a new solution with a probability min [1, exp *(-$f/T)],* where $/=/(X_0)-/(X_N) is cost reduction from the current solution X_0 to the candidate solution X_N , and *Ck* is a control parameter value at fc$^{th}$ step. *Lk* is the maximum number of solutions to be considered for each iteration number *(k).* A key point of simulated annealing is that simulated annealing accepts uphill moves with the probability exp *(-$f/Ck).* This allows simulated annealing to escape from local minima. Simulated annealing cannot cover a large region of the solution space within a limited computation time because simulated annealing is based on small moves.

The decision criterion states that accepts X_N, if either/(X_N) is less than/(X_0) (for minimisation problem) or if a random number *(u)* is less than the probability exp *(-$f/Ck*). Therefore, the simulated

annealing algorithm, besides accepting improvement in the value of the objective function, also accepts deterioration in the value of the objective function which is not so in the local search algorithm. Initially, at large value of *Ck,* large deterioration will be accepted; as *Ck* decreases, only smaller deterioration will be accepted and finally, as the value of *Ck* approaches zero, no deterioration will be accepted. Also, a large step size is taken initially and it is decreased slowly after fixed number of iterations along with the control parameter values. This procedure is continued until the control parameter reaches a specified lower limit, which can be used as a stop criterion.

## 3. GENETIC-SIMULATED ANNEALING

To improve the performance of genetic algorithm and simulated annealing, several hybrid algorithm have been proposed. Mutation operator used in genetic algorithm tends to destroy some good features of the solution at the final stages of optimisation process. Adler[10] used a simulated annealing-based acceptance function to control the probability of accepting a new solution produced by the mutation operator. Recent work on genetic algorithm-oriented hybrids are the simulated annealing genetic algorithm (SAGA) proposed by Brown", *et al.* and annealing genetic algorithm proposed by Lin[12], *et al.* Both the methods divide each generation into two phases: genetic algorithm phase and simulated annealing phase. Sirag and Weisser[13] proposed a thermodynamic genetic operator, which incorporates an annealing schedule to control the problem of applying the mutation operator. This paper proposes genetic algorithm-oriented hybrid genetic-simulated annealing algorithm method. This method is divided into two phases-genetic annealing phase and simulated annealing phase.

The hybrid algorithm incorporates the best features of genetic algorithm (searching larger regions of solution spaces) and simulated annealing (refining exhaustive solution of local region). Genetic algorithm generates a set of new solutions using the crossover/ mutation operators and then simulated annealing further refines the final best solution of genetic algorithm. The basic idea is to use the genetic operators of genetic algorithm to quickly converge the search to a near-global minima/maxima, which

**GENETIC-ANNEALING ALGORITHM**

**Genetic-annealing Phase ,**

(a) Determine, stringlett^tBMd coding scheme for the solution

p) Initialise population

(c) Evaluate population

(d) Repeat

- Apply reproduction operator
- Apply crossover operator
- Apply mutation operator
- Check for constraint satisfaction
- Evaluate population

(e) Until (termination condition)

**Simulated-annealing Phase**

(f) Select the best solution amongst the population given by last genetic pass as initial start point !

(g) Start with the best solution vector and estimate the fitness (objective function)

(h), Repeat

- , Generate a new solution in the neighbourhood and estimate its fitness
- Use an appropriate criterion for acceptance of a new move
- If the new move is accepted, make it the current move, else accept this new solution as current with a probability min (1, exp *(-$f/T))* -

(i) Until (maximum number of solutions to be considered for each iteration are performed)

(j) Reduce the temperature using a cooling schedule

(k) Repeat the steps (h) to,(j) until appropriate —stopping criteria is satisfied

**Figure 1. Pseudo code for genetic-simulated annealing**

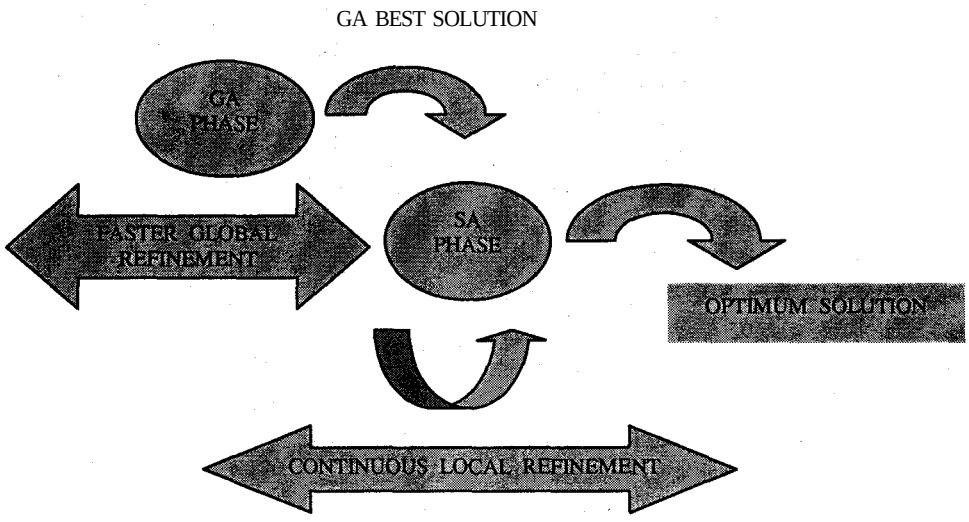


**Figure 2. Flow diagram for genetic-simulated annealing algorithm**

will further be refined to a near-optimum solution by simulated annealing using annealing process. The pseudo code for genetic-simulated algorithm algorithm is given in the Fig. 1.

In simulated annealing phase of genetic-simulated algorithm, the simulated annealing performs a biased random walk that samples the objective function in the space of independent variables. It has the ability to migrate through a sequence of local extrema in search of a global solution and to ascertain when the global extremum has been reached.

The flow diagram for genetic-simulated algorithm is given in the Fig. 2.

## 4. ANALYSIS OF GENETIC-SIMULATED ANNEALING

A concept called e-basin has been introduced, (where $e > 0$) in analysing random search algorithm. The e-basin of a solution A is a subspace in the whole solution space. It satisfies two conditions: (a) is a local optimum; (b) from any solution in the basin, a greedy search algorithm can converge to *A* with the continuing increase of solution value not greater than *e*. In other words, there are only highlands with height not greater than *e* in the basin. The heuristic behind genetic algorithm is to make the starting points generated by the genetic operator fall into the e-basins of global or good local optima. Obviously, it will take a shorter time for genetic algorithm to find a good near optimum if that is the case. This accords with the theoretical result that the rate of convergence is closely related to the starting points.

Another benefit of genetic algorithm comes from the multiple solutions it keeps during the execution. One major reason for doing this is to extend the subspace searched at the same time. The subspace searched at the same time by *m* solutions is usually greater than that searched by only one solution. The more solutions, the greater the subspace. But some method is needed to keep the subspace big and make it as big as possible. According to the concept of e-basin, multiple solutions could fall into the same basin if these are very close to each other.

Hence, these are very likely to be searched in the same area and lead to the same near optima. A simple as well as efficient way to cope with the problem is to keep the distances between any two solutions sufficiently large. Genetic operators can be used to generate two well-separated solutions if the parents come too close.

The effectiveness of keeping *m* solutions far enough apart is illustrated in the Fig. 3(a), where dashed lines represent solution positions. In

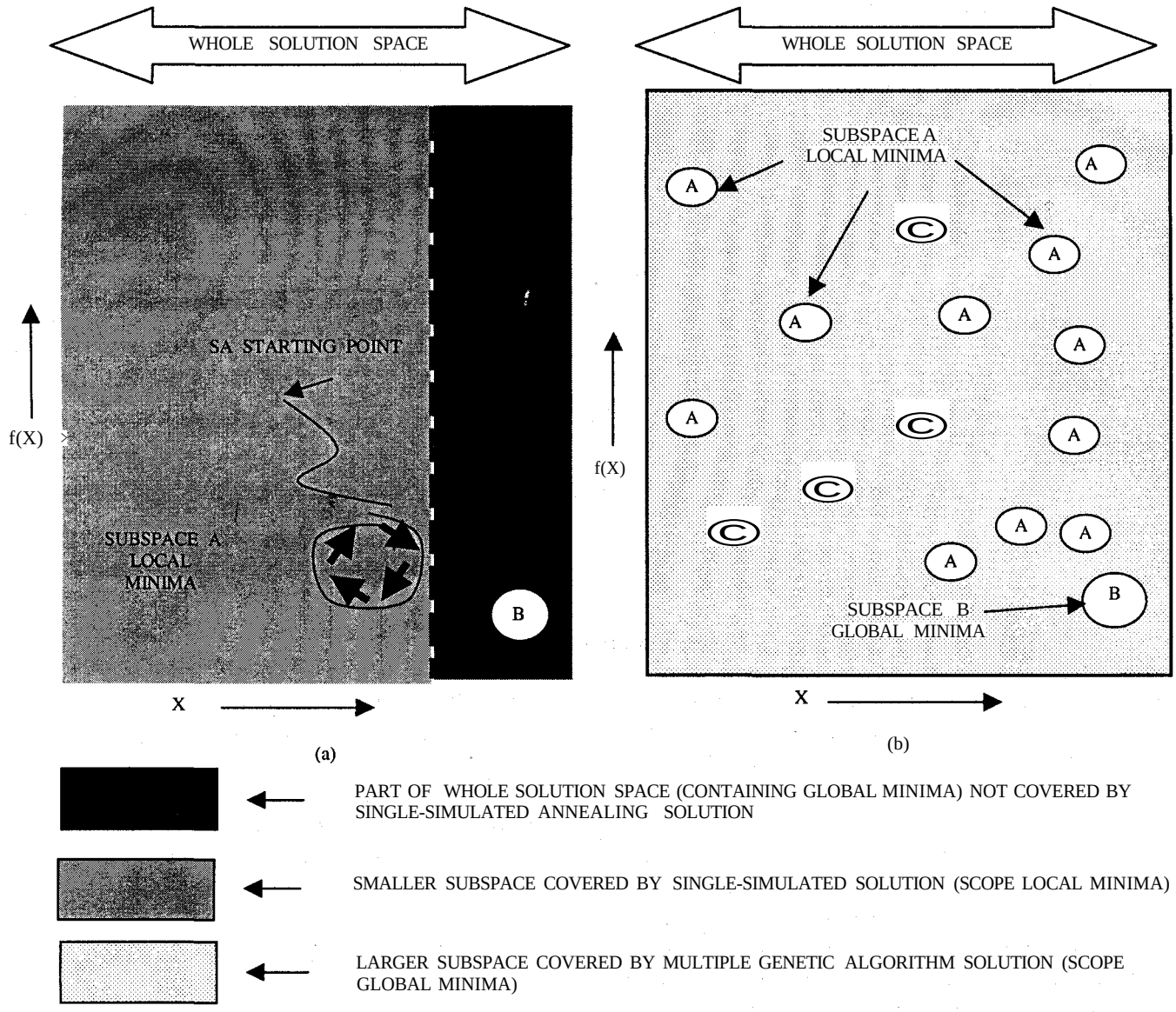


**Figure 3. Analysis of genetic-simulated annealing: (a) subspaces covered by simulated annealing and (b) subspaces covered by genetic algorithm.**

Fig. 3(a), the probability of finding the global optimum *B* is small, because the subspace searched mainly falls into the basin of local optimum *A*. If the distances between any two solutions are large, as in Fig. 3(b), the subspace searched at the same time by *m* solutions will extend to the basin of solution *B*. Hence, the global optimum *B* is more likely to be found.

The advantage of genetic algorithm becomes even greater if a parallel system is available. Genetic algorithm has a straightforward parallel implementation. *m* solutions can run on *m* processors with very small communication overhead. The genetic operations for *m* pairs of solutions can also easily be parallelised.

## 5. APPLICATION OF GENETIC-SIMULATED ALGORITHM

The new hybrid algorithm has been applied to optimal weapon allocation in multilayer defence scenario problem to arrive at a better solution than produced by the genetic algorithm or simulated annealing alone. The process of effectively allocating

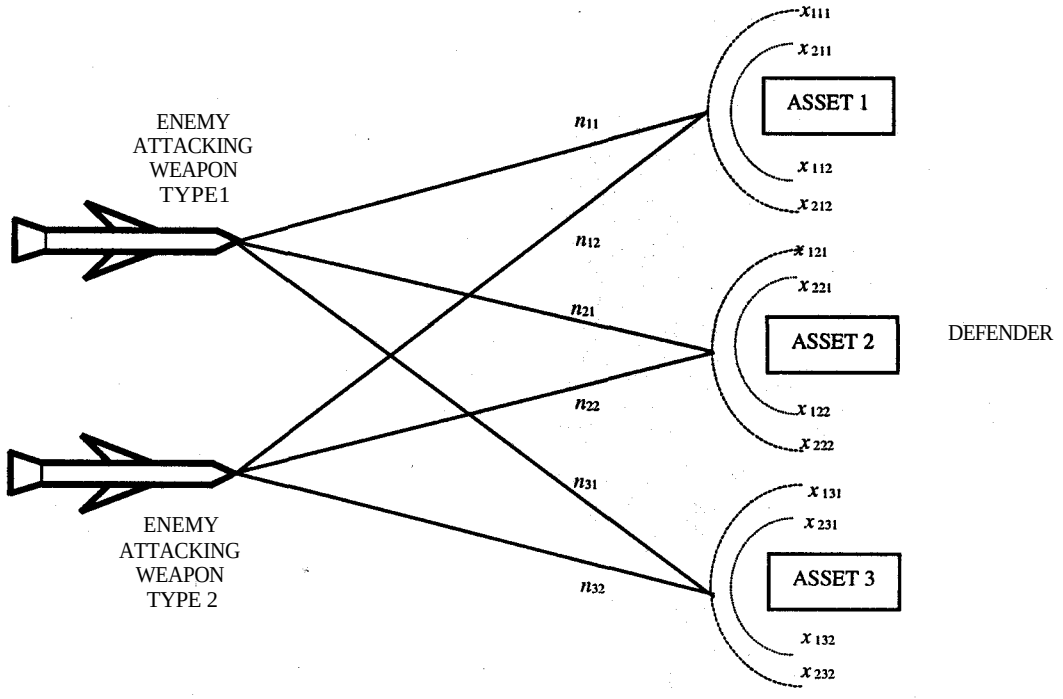


**Figure 4. Multiple layer defence**

resources (weapons in this case) against a perceived enemy threat is known as battle management/command control, and communication[6,14,15-16] ($BM/C^3$). 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, the deployment of these weapons involves their placement to protect different strategically important assets, considering 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 the 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[16].

## 5.1 Mathematical Model

For comparison, let one consider the well-established formulation[16] of the problem of a multiple layer defence in which two types of attacking weapons (Type 1, Type 2) are aimed at three different assets (Asset 1, Asset 2, Asset 3) as depicted in the Fig. 4. Two layers, each containing different types of weapons, defend these assets. Attacking weapons, which survive the interception 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 |
| $k_{rlsa}$ | 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) |
| $x_{lisa}$ | Number of defending weapons of type $d$ deployed to intercept attacking weapon of type $a$ to defend asset $s$ (defence plan) |
| $n_{ia}$ | Number of attacking weapons of type $a$ aimed at asset 5 (attack plan) |
| $g_{sa}$ | 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 $a$

$G_t$ Ground area available at asset 5

$t_d$ Ground area required by a defending weapon of type $d$

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

$M_{maxrf}$ 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, the survival probability of asset s, when attacked by all the attacking weapons of all the types is given by

$$\text{Prob(s)} = \left[ \prod_{a=1}^{A} \left[ 1 - \left\{ \prod_{d=1}^{D} (1 - k_{dsa})^{x_{dsa}/n_{sa}} \right\} g_{sa} \right] \right]^{n_{sa}} \quad (1)$$

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

$$M_{tot} = \sum_{s=1}^{S} v_s \, \text{Prob}(s) \quad (2)$$

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

$$\sum_{s=1}^{S} v_s \left[ \prod_{a=1}^{A} \left[ \text{i} - \left\{ \prod_{d=l}^{D} (1 - k_{dsa})^{x_{dsa}/n_{sa}} \right\} 0_{\&sa} \right] \right]^{n_{sa}} \quad (3)$$

It may be observed that the nonlinear objective function [Eqn(3)] has several independent parameters and the landscape is multi-modal, *ie,* 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 bound by the constraint surfaces corresponding to weapon availability, area availability, cost, and manpower. For the attacker side, the problem is to minimise the same objective function subject to the constraints imposed on the resources of the attacker against a given defence plan.

For the sake of simplicity, only weapon constraint has been considered:

$$\sum_{\text{i=1}}^{S} \sum_{\text{n=1}}^{A} x_{dsa} < Bj_d$$

for $d$ = 1, 2, ...., D

### 5.1.1. *Example Scenario*

An example similar to the one defined[15], is again considered, to compare the results. Two types of weapons are considered, which are available to defend three assets against two types of attacking weapons. Let one suppose that the maximum number of defending weapons available of the first type is 100 and that of the second type is 50. The number of attacking weapons of the first and the second types are 50 and 29. The value of the first, second, and third assets are 400, 300, and 200, respectively. The effectiveness values of defending weapons and damage probabilities of attacking weapons used for evaluating the fitness function are given in the Table 1.

While solving this problem through simulated annealing, it has been considered that the value of size of population $N$ is assumed to be 200 and is successively incremented by 100 for every increment in the value of constant $K$. The value of control parameter $T_0$ is assumed to be 0.5 and it successively decrements by 0.005 with every value of constant $K$ until it reaches 0.005. The step size delta $x$ is assumed to be 1.0 and it decrements successively by 0.1 until it reaches the value 0.1.

## 6. METHODOLOGY

Instead of taking variables themselves, the genetic algorithm works with coding of variables, which has the inherent advantage of discretising

**Table 1. Effectiveness values and damage probabilities**

| Defending weapon type (d) | Asset << | Attacking weapon type (a) | kjw | 8™ |
|---|---|---|---|---|
| 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 search space and reducing the execution timing. Generally, the representation of variables is done in binary strings. Choice of string length depends on the extent of accuracy desired. For example, in this weapon-target allocation problem in multilayer defence, all other variables being known, one has to find the number of different types of defending weapons, which will give maximum survivability to the assets.

In the problem under consideration, there are 12 different unknown $x_{rfsn}$ variables for different combinations *ofd,s,a*. Maximum number of available defending weapons of type 1 and that of type 2 is 50, ie, Bj =100, $B_2$=50;. Maximum value of these variable is 100, so the number of binary digits required to represent this maximum limit 100 is 7 ($2^7$ =128). Therefore, one needs twelve 7-bit strings to represent a set of parameters which constitute a chromosome or gene. On these chromosomes, genetic operators like crossover and mutation are applied.

A typical genome of the set of 12 parameters will look like this

| $x_{111}$ | $x_{211}$ | $x_{112}$ | $x_{212}$ | $x_{121}$ | $x_{221}$ |
|---|---|---|---|---|---|
| 0110111 | 0101010 | 0110111 | 0110111 | 0110111 | 0110100 |

| $x_{122}$ | $x_{222}$ | $x_{131}$ | $x_{231}$ | $x_{132}$ | $x_{232}$ |
|---|---|---|---|---|---|
| 0110101 | 0100111 | 0010111 | 0101101 | 0110000 | 0110101 |

## 7. RESULTS

For the problem described in this paper, the optimal defence plan using simulated annealing is given in the Table 1, which indicates that against a known attack plan[16], the maximum expected surviving

**Table 2. Optimal defence plan observed through simulated annealing and genetic-simulated annealing**

| Defending weapon type (d) | Asset << | Attacking weapon type (a), | Optimal defence plan using SA servival value 60.54 % | Optimal defence plan using GSA suvival value 63.09 % |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 2 | 1 | 1 | 0 | 4 |
| 1 | 1 | 2 | 47 | 38 |
| 2 | 1 | 2 | 0 | 2 |
| 1 | 2 | 1 | 39 | 15 |
| 2 | 2 | 1 | 14 | 26 |
| 1 | 2 | 2 | 0 | 0 |
| 2 | 2 | 2 | 5 | 3 |
| 1 | 3 | 1 | 1 | 47 |
| 2 | 3 | 1 | 16 | 0 |
| 1 | 3, | 2 | 3 | 0 |
| 2 | 3 | 2 | .15 | 15 |

**Table 3: Optimal defence plan observed through genetic-simulated annealing**

| Generatioft NO | Survivabifity | x[1][1][lj | x[1][l][2] | x[1]l2)[l] | x[l][2][2] | x[1][3]fl] | x[l][3][2] | x[2][lJU] | x[2]td][2] | x[2][2][l] | x[2][2][2] | x{2M*Hl] | xJ2][3}[2J |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| [I] | 54.60 | 11.0 | 10.0 | 11.0 | 14.0 | 15.0 | 9.0 | 4.0 | 12.0 | 9.0 | 4-0 | 9.0 | 12 |
| [2] | 56.42 | 0.0 | 30.0 | 14.0 | 5.0 | 12.0 | 5.0 | 5.0 | 6.0 | 14.0 | £6 | 5.0 | 10 |
| [3] | 56.81 | 13.0 | 15.0 | 3.0 | 10.0 | 43.0 | 8.0 | 4.0 | 12.0 | 9.0 | 5.0 | 1.0 | 13 |
| [4] | 58.03 | 15.0 | 15.0 | 3.0 | iaO | 43.0 | 8JO | 4.0 | 12.0 | 13.0 | 5.0 | 1.0 | 13 |
| *[5]* | 58.67 | 15.0 | 14.0 | 42.0 | 7.0 | 14.0 | 5.0 | 4.0 | 14.0 | 9.0 | *6JO* | *6.0* | 10 |
| [6] | 59^0 | 1.0 | 30.0 | 10.0 | 5.0 | 46.0 | 5.0 | 5.0 | 6.0 | 14.0 | 8.0 | 5.0 | 10 |
| {7] | 60.20 | 2.0 | 30.0 | 14.0 | 5.0 | 44.0 | 5.0/ | 5.0 | 6.0 | 14.0 | 8.0 | 5.0 | " |
| [8] | 60.35 | 4.0 | 30.0 | 14.0 | 5.0 | 44.0 | 1.0 | 5.0 | 4.0 | 14,0 | 8.0 | 5.0 | 14 |
| [9] | 60.50 | 2,0 | 30.0 | 14.0 | 5.0 | 44.0 | 5.0 | 5.0 | 6.0 | 15.0 | 8^ | 5.0 | 11 |
| | | | | | INITIAL INPUT TO SIMULATED ANNEALING B GENTIC ANNEALING'S BEST SOLUTION | | | | | | | | |
| SA PHASE | 60.50 | 2.0 | 30.00 | 14.0 | 5.0 | 44.0 | 5.0 | 5.0 | 6.0 | 15.0 | 8.0 | 5.0 | 11 |
| OPT SOL | 63.09 | 0 | 38 | 15 | 0 | 47 | 0 | 42 | 26 | 3 | 0 | | 15 |

value of the asset is 60.54 per cent of the total asset value. Thus, the optimal defence plan is to deploy 47 weapons of type 1 on asset 1; 39 and 19 weapons of typel and type 2 on asset 2; and 4 and 31 weapons of type 1 and type 2 on asset 3, respectively.

For the same scenario, the optimal defence plan using genetic-simulated algorithm indicates that against a known attack plan, the maximum expected survival value of the asset is 63.09 per cent of the total asset value. Thus, genetic-simulated algorithm gives the better performance than the simulated annealing alone. One of the test run analysis of this algorithm is shown in the Table 2, showing percentage survivability of each generation with value of different contributing parameters *x[d][s][a].*

In these allocations, only weapon availability constraint has been considered, but constraint on cost, manpower, and available area can also be considered. Attacker will be interested in minimising the value of the surviving assets , the same objective function can be minimised to obtain the optimal attack plan($w_{(a)}$) against a known defence plan.

## 8. CONCLUSION

Genetic-simulated annealing searches larger regions of the solution space effectively using both simulated annealing-based local search feature with genetic algorithm-based global search capability. Genetic-simulated annealing was applied to the weapon allocation problem and compared it with simulated annealing. Given the same computation resource, the experiments showed that genetic-simulated annealing improved the survival value of the assets by 3 per cent. Genetic-simulated annealing had faster rate of convergence than simulated annealing, provided mapping of the problem into population strings and fitness function is done effectively. One of the possible improvement in genetic-simulated annealing could be another approach where genetic algorithm starts with *m* solutions generated at random, and proceeds with *m* annealing processes form solutions. When all *m* annealing processes end with their near optima, a decision about whether to continue the genetic algorithm is made depending upon the quality of the solution.

The importance of genetic-simulated annealing lies not only in the particular new hybrid algorithm but also in the general principal of introducing knowledge-guided search into simulated annealing. Further work is needed to improve genetic-simulated annealing as well to develop hybrid algorithm consisting of simulated annealing and genetic algorithm.

## 9. ACKNOWLEDGEMENTS

The author is thankful to the referees for their valued suggestions.

## REFERENCES

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

2. Srinivas, M. & Patnaik, L.M. Genetic algorithms: A survey. *IEEE Comp. Mag.,* June 1994, 17-26.

3. Denning, P.J. Genetic Algorithms. *Byte,* January 1991,361-68.

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

5. Fogen, D.B. An Introduction to Simulated Evolutionary Optimisation, *IEEE Trans. Neural Networks,* 1994, 5(1), 3-14.

6. Johnson, M.E. (Ed). Simulated annealing (SA) and optimisation: Modern algorithms with VLSI, optimal design and missile defence applications. *Am. J. Mathe. Manage. ScL,* 1988, 8(3 & 4), 205-50.

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

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

9. Malhotra, Aparna & Jain, R.K. Genetic algorithm for optimal weapon allocation in multilayer defence scenario. *Def. Sci. J.,* 2001, 51(3), 285-93.

10. Adler, D. Genetic algorithm and simulated annealing: A marriage proposal. *In* Proceedings of International

Conference on Neural Network, 1993. pp. 1104-109.

11. Brown, D.; Huntley, C. & Spillane, A. A parallel genetic heuristic for the quadratic assignment problem. *In* Proceedings of 3rd International Conference on Genetic Algorithm, 1989, pp. 406-15.

12. Lin, F.T.; Kao, C.Y. & Hsu, C.C. Applying the genetic approach to simulated annealing in solving some NP^hard problems. *IEEE Trans. System, Man, Cybernetics,* 1993, 23(6), 1752-767.

13. Sirag, D. & Weisser, P. Towards a unified thermodynamic genetic operation. *In* Proceedings of 2nd International Conference on Genetic Algorithms, 1987. pp. 116-22.

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

15. Jaiswal, N.K.; Shrotri, P.K. & Nagabhushana, B.S. Optimal weapon mix, deployment and allocation problems in multiple layer defence. *Am. J. Mathe. Manage. Sci.,* 1993, 13(1&2), 53-82.

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

**Contributor**

**Mr Sanjay Bisht** obtained his MSc (Computer Science) from the Devi Ahilya Viswavidyalaya, Indore, in 1998. He joined DRDO at the Institute for Systems Studies & Analyses (ISSA), Delhi, in 1992. His areas of research include: Wargaming, heuristic optimisation and intelligent agent technology.