Evolutionary Algorithms and Simulated Annealing in the Topological Configuration of the Spanning Tree

A. SADEGHEIH
Department of Industrial Engineering
University of Yazd, P.O.Box: 89195-741
IRAN, YAZD
sadegheih@yazduni.ac.ir http://www.yazduni.ac.ir

Abstract: In this paper, the author proposes the application of a genetic algorithm and simulated annealing to solve the network planning problem. Compared with other optimisation methods, genetic algorithm and simulated annealing are suitable for traversing large search spaces since they can do this relatively rapidly and because the use of mutation diverts the method away from local minima, which will tend to become more common as the search space increases in size. Genetic algorithm and simulated annealing give an excellent trade-off between solution quality and computing time and flexibility for taking into account specific constraints in real situations. Simulated annealing is a search process that has its origin in the fields of materials science and physics. Simulated annealing, alternatively attempts to avoid becoming trapped in a local optimum. The problem of minimum-cost expansion of network is formulated as a genetic algorithm and simulated annealing. Optimal solution in linear programming is spanning tree. But GA and SA solutions show those are both spanning tree and no spanning tree.

key-words: Genetic Algorithm, Simulated Annealing, Evolutionary Algorithms, Spanning Tree, Linear Programming, Network.

1. Introduction

Network expansion is a complex mathematical optimisation problem because it involves, typically, a large number of problem variables. The commonly used methods reported in the literature can be categorised into mathematical programming, heuristic based, artificial intelligence and iterative improvement methods. In general, a characteristic of heuristic techniques is that strictly speaking an optimal solution is not sought, instead the goal is a good solution. Whilst this may be seen as an advantage from the practical point of view, it is a distinct disadvantage if there are good alternative techniques that target the optimal solution. With the development of artificial intelligence theory and techniques, some artificial intelligence-based approaches to transmission network planning have been proposed in recent years. These include the use of expert systems and artificial neural network based methods. The main advantage of the expert system based method lies in its ability to simulate the experience of planning experts in a formal way. However, knowledge acquisition is always a very difficult task in applying this method. Moreover, maintenance of the large knowledge base is very difficult. Research into the application of the artificial neural network to the planning of transmission networks is in the preliminary stages, and much work remains to be done. The potential advantage of the artificial neural network is its inherent parallel processing nature. In recent years, there has been a lot of interest in the application of simulated annealing to solving some difficult or poorly characterised optimisation problems of a multi-modal or combinatorial nature. Simulated annealing is powerful in obtaining good solutions to large scale optimisation problems and has been applied to the planning of networks, but its computational burden is very heavy. The computation speed of genetic algorithms is generally faster than that of simulated annealing. Cooling temperature could be important and neighbourhood function is crucial to its effectiveness. Simulated annealing is an

intelligent approach designed to give a good though not necessarily optimal solution, within a reasonable computation time. The motivation for simulated annealing comes from an analogy between the physical annealing of solid materials and optimisation problem. Simulated annealing simulates the cooling process of solid materials-known as annealing. However this analogy is limited to the physical movement of the molecules without involving complex thermodynamic systems. Physical annealing refers to the process of cooling a solid material so that it reaches a low energy state. Initially the solid is heated up to the melting point. Then it is cooled very slowly allowing it is to come to thermal equilibrium at each temperature. This process of slow cooling is called annealing. The goal is to find the best arrangement of molecules that minimises the energy of the system, which is referred to as the ground state of the solid material. If the cooling process is fast, the solid will not attain the ground state, but a locally optimal structure. The analogy between physical annealing and simulated annealing can be summarised as follows: i. The physical configurations or states of the molecules correspond to optimisation solution, ii. The energy of molecules corresponds to the objective function or cost function, iii. A low energy sate corresponds to an optimal solution, iv. The cooling rate corresponds to the control parameter which will affect the acceptance probability. The algorithm consists of four main components: i. Configurations, ii. Re-configuration technique, iii. Cost function and iv. Cooling schedule [1-2]. Similarly to simulated annealing, evolutionary algorithms are stochastic search methods, and they aim to find an acceptable solution where it is impractical to find the best one with other techniques.

Genetic algorithms are based in concept natural genetic and evolutionary mechanisms working on populations of solutions in contrast to other search techniques that work on a single solution. Searching not on the real parameter solution space but on a bit string encoding of it, they mimic natural chromosome genetics by applying genetics-like operators in search of the global optimum. An important aspect of genetic algorithms is that although they do not require any prior knowledge or any space limitations such as smoothness, convexity or unimodality of the function to be optimised, they exhibit very good performance in the majority of applications. They only require an evaluation function to assign a quality value to every solution produced. Another interesting feature is that they are inherently parallel, therefore their implementation on parallel machines reduces significantly the CPU time required. From the above review, the following conclusions are drawn regarding previous methods: **i.** most require a large number of decision variables; **ii.** long computation times; **iii.** most allow no user interaction; **iv.** models fixed by program formulation; **v.** considerable effort and good mathematical knowledge is usually required for adaptation to specific problems [3-7].

The work presented here was carried out using the Microsoft^{R} Excel^{TM} spreadsheet and an add-in to provide the GA. This add-in is called Evolver^{TM} and is developed and supplied by Axcelis [18]. The model of network planning developed in this research is built in Excel^{TM} using the spreadsheet's built-in functions. After building the model, the GA is run to optimise the network given an objective function. The fitness value and decision variables are passed back to the GA component which is independent of the spreadsheet model. At the end of the GA run, when the stopping criterion is met, the best network is presented in a tabular form in the spreadsheet. The chromosome structure used to represent a particular set of possible transmission line power capacities, for the mixed-integer transmission network planning using GA has nine state variables. Each individual line capacity is encoded by sufficient bits to cover its allowable range of values. The initial population is generated randomly, that is, each bit in each chromosome is set randomly to either 1 or 0. Whenever a new chromosome is generated it is checked to see that in decoded form it produces valid values for the genes. When an invalid value is produced the chromosome is discarded and another one is generated. The spreadsheet model is developed for solving this problem. In the next step for solving the system planning using a GA, equations must be satisfied.

The final step in the implementation of the system planning using a GA is the fitness function. The fitness value of a chromosome is a measure of how well it meets the desired objective [19-20]. In this case the objective is the minimisation of the network's cost function. Choosing and formulating an appropriate objective function is crucial to the efficient solution of any given genetic algorithm problem. When designing an objective function for an optimisation problem with constraints, penalty functions can be introduced and applied to individuals that violate the imposed constraints.

The results from a set of tests carried out on the prototype show that the application of genetic algorithm techniques and simulated annealing are feasible in network planning. A novel empirical analysis of the effects of the algorithm's parameters is also presented in the context of this spanning tree application. The goal is to determine which new line to construct in order to supply the future load pattern. The transmission network synthesis used here is based on the linearized power-flow model [8].

2. Genetic Algorithm

One of the most popular evolutionary algorithms developed so far is genetic algorithms. Each string represents a possible solution to the problem being optimized and each bit represents a value for some variable of the problem. These solutions are classified by an evaluation function, giving better values, or fitness, to better solutions. Each solution must be evaluated by the fitness function to produce a value. The selection operator creates a new population by selecting individuals from the old population, biased towards the best. Crossover is the main genetic operator and consists of swapping chromosome parts between individuals. The last operator is mutation and consists of changing a random part of the string. Mutation works as a kind of life insurance. Some important bit values may be lost during selection and mutation can bring back, if necessary. Nevertheless, too much mutation can be harmful: a mutation probability of near one always leads to random search [17], independently of crossover probability. Inversion is an operator which takes a random segment in a solution string and inverts it end to end. See Table 1. Unlike crossover and mutation which occur in nature, inversion is an entirely artificial operator with the sole purpose of producing a new gene arrangement. The inversion rate is the probability of performing inversion on each individual during each generation. It is normally set between 12% and 30%.

Table 1: Inversion points (**) and newly inverted and mutated in spring 1

String 1	0	0	0	**1	1	0	1**	1	0
String 1	0	0	0	0	0	1	0	1	0

Genetic algorithms are heuristic search techniques that utilize an analogy to "survival of the fittest". Genetic algorithms employ a population of solutions and, through the algorithm's main operator of crossover, combine parts of good solutions in an attempt to create better solutions. Genetic algorithms success on variety of problem has led to their application in many fields. Genetic algorithms are essentially search algorithms based on the mechanics of genetics, natural selection and survival of the fittest. Genetic algorithms are efficient search procedures that can identify optimal or near optimal solutions in solution populations. The

adaptive search strategy maintains a population of individual structures in encoded form, referred to as artificial chromosomes that represent potential solutions in the space being searched. The genetic algorithm iterates for a specified number of generations or until a measure of convergence is observed in the "best" solution. At each generation the chromosome population is subjected to selection, crossover and mutation operations. The selection of a chromosome is based on the fitness of the chromosomes in the population. This operator can be implemented in a variety of ways. In this paper parents are chosen according to the rank-based mechanism. The advantage of rank-based is that it prevents too quick convergence of the GA. The crossover operator takes two chromosomes and produces an offspring by exchanging sections of the chromosomes with each other. The crossover operator allows for partial exchange of information between the chromosomes. The different crossover operators for recipe genetic algorithms have been in the literature. Uniform crossover generally works better than one and two-point. It has been shown that in almost all cases, uniform crossover is more effective at combining schemata than either one or two point crossover. Empirically, uniform crossover been shown to be more effective on a variety of function optimisation problems. The crossover operator used in the genetic algorithm in this paper is the uniform crossover. The mutation gives random movement about the search space thus preventing the GA becoming trapped in "local optima" or "dead corner" during the search. In this paper a random number is generated for each gene in the chromosome, if the random number generated is less than or equal to the mutation rate then gene is mutated otherwise not. In each generation relatively "good" solutions reproduce, and relatively "bad" solutions die, to be replaced by offspring of the good. To distinguish between solutions, an evaluation or objective function plays the role of the environment. It is generally accepted that any genetic algorithm to solve a problem must have five components: **i.** a genetic representation of solutions to the problem, **ii.** a way to create an initial population of solutions, **iii.** an evaluation function that plays the role of the environment, **iv.** genetic operators that effect the composition of children during reproduction, and **v.** values for the parameters that the genetic algorithm uses. In general, each of these components affects the genetic algorithm performance significantly.

There has been a big growth in their application and development since the mid-1980s when the practical application for genetic algorithms became apparent. GA has been successfully applied in various areas such as computer science, engineering and operations research. The strength of genetic algorithms is that they are free from limitations about the search space, e.g., continuity, differentiability and unimodality and they are very flexible in the choice of an objective function. Furthermore, genetic algorithms can work on very large and complex spaces. These properties give genetic algorithm the ability to solve many complex real-world problems and transmission network planning in particular. Comparison of the formulaic with the previous mathematical programming formulaic shows a significant simplification due to the " power in line" variables.

3. Computational results

Genetic algorithm which works by: **i.** Emulating the natural process of evolution. **ii.** Means of progressing toward the optimum solution. **iii.** Starts with an initial set of random configurations, called population (each individual in the population is a string of symbols, usually a binary bit string representing a solution). **iv.** Each iteration, called generation, the individuals in the current population, are evaluated – fitness value – fitter individuals have a higher probability of being selected. **v.** To generate new individuals called offspring, by using genetic operators. **vi.** The offspring are next evaluated, and new generation is formed by selecting some of the parents and offspring, once again on the basis of their fitness. A binary chromosome consists of binary digits. A 1 or 0 in the string may, in a Boolean scheme, correspond to whether some condition is true or false, or bits may be strung together to form binary words that will translate either directly or

indirectly into continuous valued variables. The first step in the application of a GA is the coding of the variables that describe the problem. The most common coding method is to transform the variables to a binary string of specific length. This string represents the chromosome of the problem and the length of the chromosome represents the number of zeros and ones in the binary string. By decoding the individuals of the initial population, the solution for each specific instance is determined and the value of the objective function that corresponds to this individual is evaluated. This applies to all members of the population. When it comes to reproduction a GA may operate in a generation mode or in a 'steady-state' mode. In generation mode each iteration of the GA produces a whole new generation of chromosomes. In contrast, the steady-state GA produces, at each iteration, just one new 'child' chromosome from two selected parents. This child is added to the existing population and the least fit member of the population is then deleted to maintain the population size. The steady-state GA is used in this paper. The advantage of using steady-state reproduction is that all the genes are not lost as is the case in generational replacement where after replacement many of the best individuals may not be produced at all, and their genes may be lost. Steady-state reproduction is a better model of what happens in longer-lived species in nature. This allows parents to nurture and teach their offspring, but also gives rise to competition between them. A rank based method is used here. The members are ranked in order of their fitness and the probability of selection is inversely rated to this ranking. The advantage of a rank based approach is that it helps to avoid too rapid a rate of convergence that may lead to the population being swamped by a local optimum due to the loss of diversity. Author proposed uniform crossover which works as follows. Two parents are selected, and two children are produced. For each bit position on the two children, it is decided randomly as to which parent contributes its bit value to which child. In the approach, several GA parameters such as population size and genetic operator probability are included. These parameters may influence the performance. In this problem, combination of following GA parameters is tested. i. Population Size (10, 50, 90), ii. Crossover Rate (0.1, 0.3, 0.5, 0.6, 0.9) and iii. Mutation Rate (0.001, 0.006, 0.011, 0.2). For each combination of the parameters the GA is run for eight different random initial population. These eight population are different for each combination. Thus, in total, the GA is run four hundred and eighty times. The values chosen for the population size are representative of the range of values typically seen in the literature, with 0.016 being included to highlight the effect of a relatively large mutation range. The crossover rates chosen are representative of the entire range. Distribution of number of iterations required to analyse the distribution of the number of trials required to reach the optimal solution. Iterations show that the number of iterations they require is a very good fit to log-normal distribution. This means that statistical analysis techniques that assume normality can be applied to log of the number of iterations required. Meanwhile, combinations are tested by analysis of variance techniques using the F-test, in order to examine relation between the parameters and the performance. Results which are observed in this experiment are as follows: **i.** the crossover rate is an insignificant factor in the performance of GA, but crossover rate of which range from 0.1 to 0.9 is to be desirable on whole. **ii.** in Tables 2, 3, 4 and 5, an effect of mutation rate on the GA approach is displayed in case that crossover rate and population size are fixed. There is a degree of insensitivity to mutation rate (in this research, mutation rate of which range from 0.006 to 0.011 are suitable) in so far as good value form a fairly broad range rather than coming at a more precise point. However, outside of the good range performance soon deteriorates greatly that is significant. And also it is observed that there are trade off relation between a mutation rate and a population size. That is, low mutation rate is desirable if population size is large, and high mutation rate is desirable if population size is small. The sensitivity to population size is greatly reduced when the mutation rate is in the good range and is an insignificant. However, outside of the good range is significant.

Table 2: Crossover rate = 0.5 and population size = 50 and mutation rate = 0.001

Fitness Function($\times 10^4$)	0.384	0.434	0.476	0.543	0.549
Iterations	0	2000	2450	5500	7000

Table 3: Crossover rate = 0.5 and population size = 50 and mutation rate = 0.006

Fitness Function($\times 10^4$)	0.500	0.543	0.555	0.561	0.563
Iterations	0	500	2500	4700	5500

Table 4: Crossover rate = 0.5 and population size = 50 and mutation rate = 0.011

Fitness Function($\times 10^4$)	0.300	0.400	0.526	0.555	0.563
Iterations	0	1000	2000	3800	5500

Table 5: Crossover rate = 0.5 and population size = 50 and mutation rate = 0.20

Fitness Function($\times 10^4$)	0.400	0.454	0.512	0.529	0.555
Iterations	0	3000	4000	6000	7000

4. Simulated annealing

Physical annealing refers to the process of cooling a solid material so that it reaches a low energy state. Initially the solid is heated up to the melting point. Simulated annealing is an intelligent approach designed to give a good though not necessarily optimal solution, within a reasonable computation time. The motivation for simulated annealing comes from an analogy between the physical annealing of solid materials and optimisation problem. Simulated annealing simulates the cooling process of solid materials-known as annealing. However this analogy is limited to the physical movement of the molecules without involving complex thermodynamic systems. Physical annealing refers to the process of cooling a solid material so that it reaches a low energy state. Initially the solid is heated up to the melting point. Then it is cooled very slowly allowing it is to come to thermal equilibrium at each temperature. This process of slow cooling is called annealing. The goal is to find the best arrangement of molecules that minimises the energy of the system, which is referred to as the ground state of the solid material. If the cooling process is fast, the solid will not attain the ground state, but a locally optimal structure. Similarly to simulated annealing, evolutionary algorithms are stochastic search methods, and they aim to find an acceptable solution where it is impractical to find the best one with other techniques. The simulated annealing algorithm was employed to solve the problems of combinatorial optimisation of network systems. Simulated annealing is a hill-climbing algorithm used to find near-optimal solution. It works in the principle of annealing of the material, which searches for the most convenient energy state. It starts with an initial configuration that is achieved with any other algorithm. New configurations are generated by applying movements, which are randomly selected with a defined probability. A high temperature is given initially, when all movements are accepted even if they increase the energy. As temperature

decreases, fewer movements are accepted, unless they improve the energy. Finally, at very low temperatures, only movements that improve the energy are accepted [9-16].

Simulated annealing algorithm:

Temp=Start Value;

S(old)=Initial Solution;

E(old)=Energy[S(old)];

While *Temp* $\succ$ *Temp(end)* **do**

While *Change()* **do**

S(new)=Adjust[S(old)];

E(new)=Energy[S(new)];

ΔE = *E(new)-E(old);*

x=F (ΔE *, Temp);*

r= Random (0, 1);

if $r \prec x$ **then**

S(old)=S(new);

E(old)=E(new);

Endif

Endwhile

Temp=Update(Tem);

Endwhile

The same 10-bit binary string representation scheme applied in the case of the genetic algorithm was implemented for simulate annealing because of its flexibility and ease of computation. The cost function for this problem is objective function given in equation (1).

Minimise: $Z = \sum_{NP} K_{ij} \left| P_{ij} \right|$ (1)

In this case the objective is the minimisation of the network's cost function, so the fitness value is the reciprocal of this cost. The annealing process stared at a high temperature, T=2000 units, so most of the moves were accepted. The algorithm was implemented in Turbo C++. The initial stopping criterion was set at a total unit cost of optimal solution found by the genetic algorithm (0.563). Eight cooling rates were used (0.45, 0.50, 0.55, 0.80, 0.85, 0.90, 0.95, 0.97). The final cost function is showed in Table 6, and Fig.s 1, 2 that the maximum number of iterations was set at 7000.

Table 6: Fitness function and iterations

Cooling Rate	0.45	0.50	0.55	0.80	0.85	0.90	0.95	0.97
Fitness Function($\times 10^4$)	0.521	0.533	0.563	0.563	0.563	0.555	0.534	0.540
Iterations	3000	2900	6700	5800	6501	7000	7000	7000

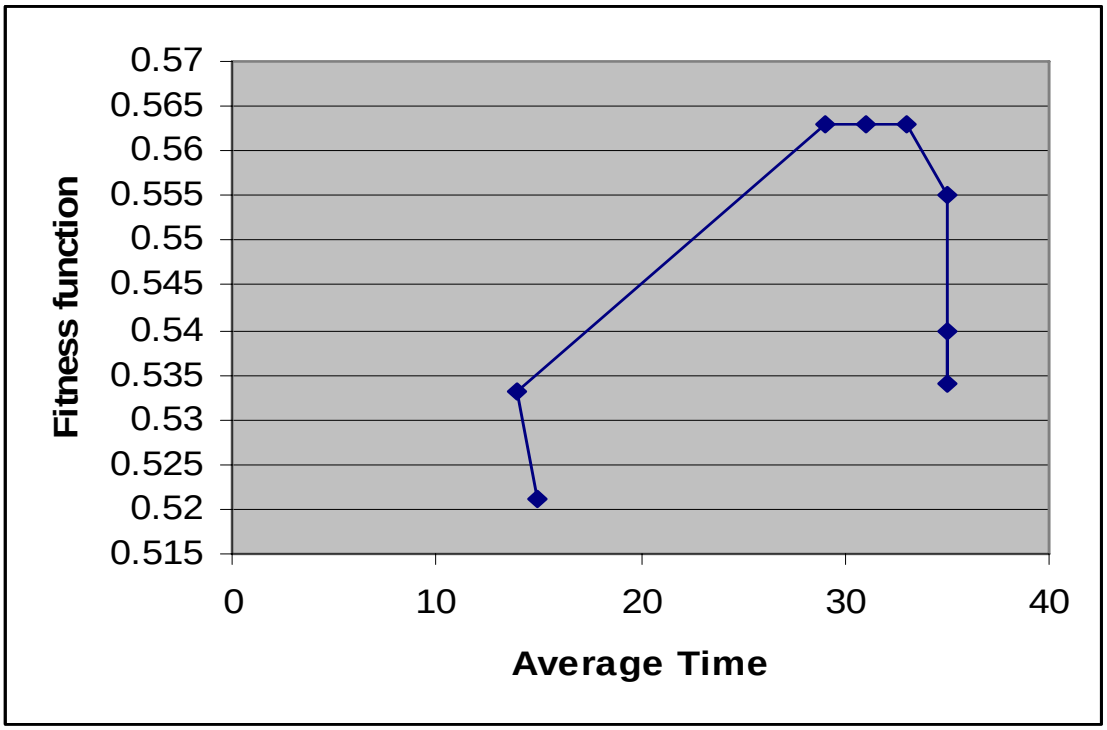

Figure 1: Average time and fitness function

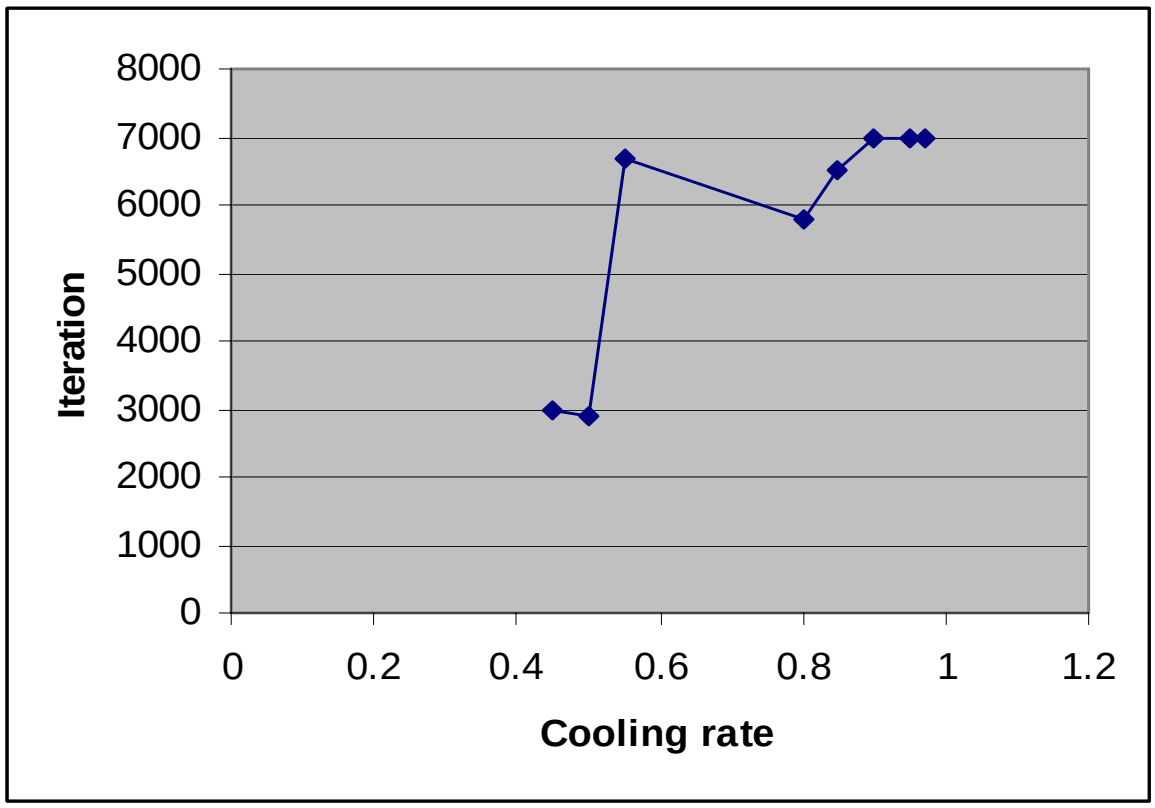

Figure 2: Cooling rate and Iterations

Therefore the simulated annealing algorithm obtained a value of 0.563. Optimal solution in linear programming is spanning tree (a spanning tree is a connected subset of a network including all nodes and containing no loops) that is the fundamental theorem for the network simplex method. But GA and SA solutions show those are both spanning tree and no spanning tree. These results demonstrate the validity and effectiveness of the proposed methodology and flexible and straightforward methods. Genetic algorithm is able to produce optimal solution by simulating the adaptive nature of natural genetics. The results presented here support the extension of this argument into the field of topological configuration of network system in particular.

There are several procedures used to generate an initial basic feasible solution. For example:

i. The Northwest-Corner Method

STEP (1) move to the cell in the upper left-hand corner (cell (i, j) in the table denotes the flow allocation X_{ij} from source i to destination j);

STEP (2) allocate to the current cell the maximum feasible amount, considering the availability and requirement constraints;

STEP (3) if no move cells to right or down STOP. If units left to allocate for the current supply move one cell to the right, else move one cell down. Go to STEP (2).

ii. The Minimum Matrix Method

An alternative procedure, which is cognisant of the costs and straightforward to implement, is the minimum matrix method. This method iteratively allocates as much as possible to the available arc with the lowest cost.

iii. Vogel's Approximation Method

STEP (1) use an extended form of the transportation table of the problem to calculate and record the arithmetic difference between the second smallest and the smallest cost (for the maximisation problem between the highest and the second highest profit) for each row and column. Call these the row and column differences respectively;

STEP (2) select the row or column that has the largest difference;
STEP (3) allocate as much as possible to the smallest cost cell in that row or column (for a maximisation problem, to the largest profit cell) and show the allocated number in a circle;
STEP (4) cross out the cost figures in the row or column where the supply or demand is exhausted;
STEP (5) do not consider those columns and rows which have been fully allocated. Stop if no rows and columns remain; otherwise, go to STEP 1.
The three procedures for finding an initial basic feasible solution result in different bases; however, all have a number of similarities. Each base consists of exactly (m+n-1) arcs, one less than the number of nodes in the network. Further, each base is a sub-network that satisfies the following two properties:
i. every node in the network is connected to every other node by a sequence of arcs from the sub-network, where the direction of the arcs is ignored;
ii. the sub-network contains no loops, where a loop is a sequence of arcs connecting a node back to itself and the direction of the arcs is ignored.
A sub-network that satisfies these two properties is called a spanning tree.

5. Conclusions

Genetic algorithms are different from traditional optimisation and search procedure in following ways. **i.** they work with a symbolic representation of solution parameters rather than the parameters themselves, **ii.** they treat a set of potential solutions, not a single solution, **iii.** they use only information about the pay off (or the objective function), not any other auxiliary information, **iv.** they use probabilistic transition, not deterministic rules. In this paper, It introduces a genetic algorithm and a simulated annealing method for the topological configuration of the network system. Simulated annealing simulates the cooling process of solid materials-known as annealing. However this analogy is limited to the physical movement of the molecules without involving complex thermodynamic systems. Physical annealing refers to the process of cooling a solid material so that it reaches a low energy state. Initially the solid is heated up to the melting point. Then it is cooled very slowly, allowing it is to come to thermal equilibrium at each temperature. GA and SA solutions are showed those are both spanning tree and no spanning tree and the decision variables for solving GA and SA are less than linear programming. These results show those genetic algorithm and simulated annealing techniques are feasible in the network planning.

References

[1] P. J. M. van Laarhoven and E. H. L. Aarts, “Simulated annealing: Theory and applications”, Reidel, Dordrecht, Holland, 1987.

[2] P. Tian and Z. Yang, “An improved simulated annealing algorithm with genetic characteristics and travelling salesman problem”, Journal of Information and Optimisation Sciences, Vol. 14, No. 3, 1993, pp. 241-255.

[3] A. Sadegheih, “Scheduling problem using genetic algorithm, simulated annealing and the effects of parameter values on GA performance”, Applied Mathematical Modelling, Issue 2, Vol. 30, Feb. 2006, pp. 147-154.

[4] A. Sadegheih and P. R. Drake, “Network Planning using Iterative Improvement and Heuristic Method”, International Journal of Engineering, Vol. 15, No. 1, 2002, pp. 63-74.

[5] A. Sadegheih, “New formulation and analysis of the system planning expansion model’, European transactions on Electrical Power, Accepted, Wiley InterScience, Vol.17, pp. 1-18, 2007.

[6] A. Sadegheih, “Global optimisation using evolutionary algorithms, simulated annealing and tabu search”, WSEAS Transactions on Information Science and Applications, Issue 6, Vol.1, Dec. 2004, pp. 1700-1706.

[7] A. Sadegheih, “Models in the iterative improvement and heuristic methods”, WSEAS Transactions on Advances in Engineering Education, Issue 4, Vol. 3, April 2006, pp. 256-261.

[8] R. Villasana, L. L. Garver and S. J. Salon, “Transmission network planning using linear programming”, IEEE Trans. on PAS, Vol. PAS-104, No. 2, Feb. 1985, pp. 349-356.
[9] A. Sadegheih “Design and implementation of network planning system”, 20th International Power System Conf. 14-16, Nov., 2005, Tehran, Iran.
[10] A. Sadegheih, “Modelling, simulation and optimisation of network planning methods”, Proceeding of the WSEAS Int. Conference on Automatic Control, Modeling and Simulation, Prague, Czech Republic, March 12-14, 2006.
[11] A. Sadegheih, “Sequence optimization and design of allocation using GA and SA”, Applied Mathematics and Computation, Issue 2, Vol. 186, pp. 1723-1730, 2007.
[12] A. Sadegheih, "A novel method for designing and optimization of network", International Journal of Engineering, Vol. 20, No. 1, pp. 17-26, 2007.
[13] A. S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, “Optimization by simulated annealing”, Science, Vol. 220, No. 4598, 1983, pp. 671-680.
[14] R. Romero, R. A. Gallego and A. Monticelli, “Transmission system expansion planning by simulated annealing”, Proc. 1995 IEEE Power Industry Computer Application Conference (PICA’95), USA, pp. 278-283.
[15] F. Albuyeh and J. J. Skiles, "A transmission network planning method for comparatives studies", IEEE Trans. on PAS, Vol. PAS-100, No. 4, pp. 1679-1684, 1981.
[16] A. O. Ekwue, “Investigations of the transmission system expansion problem”, Electric power and energy systems, Vol. 6, No. 3, pp. 139-142, 1984.
[17] D. E. Goldberg, Genetic algorithm in search, optimization and machine learning”, Addison-Wesley, Reading, MA, 1989.
[18] “Evolver user’s guide” Axcelis, Inc. Seattle, WA, USA, 2005.
[19] J. E Baker, “Adaptive selection methods for genetic algorithms”, In J. J. Grefenstette, ed., Proceedings of First International Conference on Genetic Algorithms, Erlbaum, 1985.
[20] L. Davis, “Handbook of Genetic Algorithms”, Van Nostrand Reinhold, New York, 1991.