Using the invariant optimal assignment of a *k-out-of-n : G* system to test the Effectiveness of Genetic Algorithms

Dr Leow Soo Kar
School of Information Technology
Monash University Malaysia
No 2 Jalan Kolej, Bandar Sunway
46150 Petaling Jaya, Malaysia
leow.soo.kar@infotech.monash.edu.my

Koh Sue-Yi
School of Information Technology
Monash University Malaysia
No 2 Jalan Kolej, Bandar Sunway
46150 Petaling Jaya, Malaysia
sue_street@hotmail.com

Abstract – **Genetic algorithms have been used to find solutions to difficult optimization problems which are not amenable to exact methods or whose solution times are just impractical. GA's are generally successful in finding a solution or solutions which are normally termed as near optimal. The quality of the GA-produced solution is usually measured by comparing the GA solution with the known optimal solution of small to medium size instances of the problem. For large instances of the problem, this quality can only be estimated subjectively. In this paper, we use a non-linear assignment problem whose size can be made arbitrarily large but whose optimal solution is theoretically known, to test the effectiveness of the genetic algorithm.**

I. INTRODUCTION

Genetic algorithms [1] have been applied to a wide variety of problems and they are known to be a good approach to solving NP-Complete problems [2]. They are also very good heuristic methods to find near optimal solutions in acceptable solution times. Like all heuristic methods, the quality of the GA solution cannot be judged except for small to medium size problems.
In this paper, we use a problem whose size can be made arbitrarily large but whose optimal solution is theoretically known, to test the performance of a genetic algorithm. A problem with such characteristics is presented and described below.

Given two sets of sub-components with different reliabilities:

$$C = \{ c_1, c_2, c_3 \dots\dots, c_n \}$$
$$D = \{ d_1, d_2, d_3 \dots\dots, d_n \}$$

We define a component of the system to be a pairing of sub-components in set C to those in set D. The pairing of the sub-components is called a component of the system and is said to be functioning correctly if both sub-components are functioning correctly.

A *k-out-of-n:G* system will consist of n such components and have at least *k* of them working correctly.

The problem of assigning these sub-components from one set to those of a second set, to maximize system reliability for a *k-out-of-n:G* system was first investigated in [3] as a fault tolerant computer system. Leow [3] formulated the problem as a non-linear assignment problem and they showed that the optimal assignment of such a system was invariant for any value of *k* and *n*.

In this paper, we use this result to reverse the roles of the solution method and the problem to be solved. Instead of using GAs to solve the problem, we use the problem to evaluate the effectiveness of the method. The performance metric of the GA is the ability to reach an optimal assignment. The structure of this paper is as follows: Section II presents the general formulation of the *k-out-of-n : G* system and Section III describes the genetic algorithm used for the experiments. In Section IV we present the experimental results of our test cases and finally in Section V, we draw some conclusions.

II. THE K-OUT-OF-N : G SYSTEM

The reliability problem of the *k-out-of-n : G* system can be formulated as an assignment problem as follows :
Instead of maximizing system reliability, we will minimize system unreliability and we will also assume that the individual reliabilities of the subcomponents can be ordered as follows:

$$c_1 \geq c_2 \geq c_3 \geq \dots\dots\dots\dots \geq c_n$$
$$d_1 \geq d_2 \geq d_3 \geq \dots\dots\dots\dots \geq d_n$$

The following notations are used in the formulation:

$I : \{1,2,3\dots\dots\dots n\}$

$C_S(I)$: set of all distinct combinations of I taken s at a time

$$\text{Let} \quad x_{ij} = \begin{cases} 1 & \text{if subcomponent } i \text{ is assigned to } j \\ 0 & \text{otherwise} \end{cases}$$

Min

$$z(x) = \prod_{i=1}^{n}\prod_{j=1}^{n}(1 - c_i d_j)^{x_{ij}} +$$

$$\sum_{s=1}^{k-1}\sum_{\alpha \in C_s(I)}\left\{\prod_{j=1}^{n}(c_{\alpha(1)}d_j)^{x_{\alpha(1),j}}(c_{\alpha(2)}d_j)^{x_{\alpha(2),j}}\ldots(c_{\alpha(s)}d_j)^{x_{\alpha(s),j}}\right\}\times$$

$$\left\{\prod_{j=1}^{n}(1-c_{\beta(1)}d_j)^{x_{\beta(1)j}}(1-c_{\beta(2)}d_j)^{x_{\beta(2)j}}\ldots(1-c_{\beta(n-s)}d_j)^{x_{\beta(n-s)j}}\right\} \tag{1}$$

where $\alpha = \{\alpha(1), \alpha(2), \ldots\ldots\ldots\ldots \alpha(s)\} \in C_S(I)$ and
$\beta = \{\beta(1), \beta(2), \ldots\ldots\ldots\ldots \beta(n\text{-}s)\}$
= complement of α in *I*.

subject to:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,2,\ldots\ldots n \tag{2}$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,2, \ \ldots n \tag{3}$$

$$x_{ij} \in \{0,1\}$$

The first term in (1) is the probability of all components failing and the second term is the sum of the probabilities of 1,2,3,... (k-1) components functioning correctly.

For example, the *2-out-of-n : G* system is formulated as follows:

$$\text{Min } z(x) = \prod_{i=1}^{n}\prod_{j=1}^{n}(1 - c_i d_j)^{x_{ij}} +$$

$$\sum_{i=1}^{n}\left\{\prod_{j=1}^{n}(c_i d_j)^{x_{ij}}\prod_{\substack{s=1\\ s\neq i}}^{n}\prod_{j=1}^{n}(1-c_s d_j)^{x_{sj}}\right\}$$

subject to:

$$\sum_{i=1}^{n} x_{ij} = 1 \qquad j = 1,2,\ldots\ldots n$$

$$\sum_{j=1}^{n} x_{ij} = 1 \qquad i = 1,2, \ \ldots\ldots n$$

$$x_{ij} \in \{0,1\}$$

III. GENETIC ALGORITHMS

Genetic algorithms (GAs) are heuristic methods that have been used on many difficult optimization problems and been found to produce 'good', near-optimal or sometimes even optimal solutions. But, the effectiveness of the GAs is normally assumed. Many researchers have used various function groups to test and study the performance of GAs [4,5]. They have focused mainly on tweaking the GA control parameters and operators to enhance the performance. In this paper, we use a problem which can be made arbitrarily large but whose optimal solution is theoretically known, to conduct experiments on the effectiveness of GAs, i.e. the ability of the GA to reach an optimal solution.

A simple GA was written in Visual C++ to solve this problem. Its properties are as follows:

- Steady state GA.
- Population size is fixed at 100.
- Selection of individuals for each crossover is done by randomly selecting 2 individuals from the parent population.
- Number of crossovers performed is a direct factor of the crossover rate of 0.2.
- Resulting child population allows for duplicates.
- Selection of individuals for mutation is done by randomly selecting a single individual at each time.
- Mutation function is applied to the resulting child population obtained after the process of crossover during each generation run.
- No. of individuals selected for mutation is a direct factor of the mutation rate of 0.1.
- Mutation occurs by randomly picking 2 alleles from each individual and switching them.

IV. EXPERIMENTAL RESULTS

Using the GA described in the previous section, we conducted a number of experiments to test the effectiveness of the GA, i.e. its ability to find the optimal assignment. We chose to experiment on a 2-out-of-n:G system and as previously mentioned, the problem sizes can be made as large as we liked. We partitioned the problems into 3 sets:

Small problems with sizes starting from 10 subcomponents to 100.
Medium problems with sizes starting from 110 to 200 subcomponents.
Large problems with sizes larger than 210 to 300 subcomponents.

Fifty cases of each problem size were randomly generated and the GA was allowed to run up to 50,000 generations for small and medium problems while a 100,000 generations were set for large problems. Table 1 shows the success rate (i.e. percentage of cases where optimality was reached) and the average number of generations needed to reach optimality for each of the problem sizes.

The GA performed very well and was able to achieve 100 percent success rate in all three problem sets. Table 1 shows the average number of generations needed to reach optimality for each of the problem sizes. Chart 1 plots the problem size against the average number of generations required to reach optimality and it shows that the number of generations is of order $o(n^2)$. We can deduced from here that the GA was relatively fast in reaching the optimal solution.

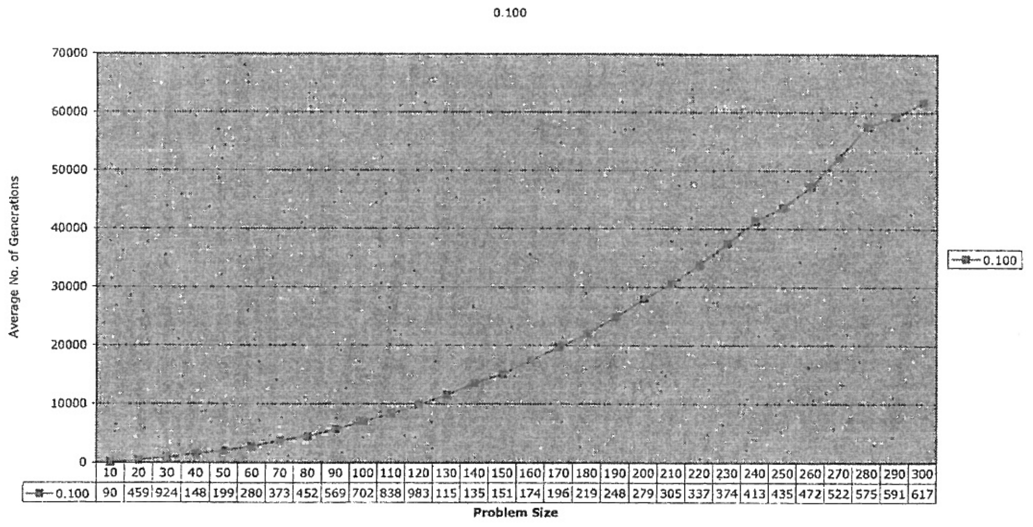
V. CONCLUSIONS

GAs, as have been mentioned before, provide a very good heuristic method for a wide variety of optimization problems but the quality of the GA solutions cannot be ascertained very accurately except for small to medium size problems. In this paper, we used a set of test problems whose optimal solution is known and whose size can be arbitrarily large to test the effectiveness of algorithm. The experimental results suggested that GAs can be a robust and fast heuristic method for solving large and complex problems.

TABLE 1
Success Rate and Average Number of Generations for Small, Medium and Large Problems

Small Problems			Medium Problems			Large Problems		
Problem size	Success Rate %	Average number of generations	Problem size	Success Rate %	Average number of generations	Problem size	Success Rate %	Average number of generations
10	100	90	110	100	8388	210	100	30544
20	100	459	120	100	9832	220	100	33772
30	100	924	130	100	11520	230	100	37492
40	100	1480	140	100	13539	240	100	41368
50	100	1998	150	100	15138	250	100	43582
60	100	2803	160	100	17446	260	100	47218
70	100	3736	170	100	19671	270	100	52296
80	100	4520	180	100	21981	280	100	57510
90	100	5693	190	100	24822	290	100	59177
100	100	7023	200	100	27902	300	100	61788

CHART 1
Problem Size Vs Average Number of Generations



	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180	190	200	210	220	230	240	250	260	270	280	290	300
0.100	90	459	924	148	199	280	373	452	569	702	838	983	115	135	151	174	196	219	248	279	305	337	374	413	435	472	522	575	591	617

VI. REFERENCES

[1] J. H. Holland, "*Adaption in natural and artificial systems*", Ann Arbor: The University of Michigan Press, 1975.

[2] Kenneth A De Jong, William M Spears, "Using genetic algorithms to solve NP-complete problems" *Proceedings of the third international conference on Genetic Algorithms"* Morgan Kaufmann Publishers Inc. San Francisco, CA, USA.

[3] Leow Soo Kar, "Maximum Reliability for Combined Hardware-Software Fault-Tolerant Computer Systems", *OPSEARCH*, Vol. 27, No 1,1990, pp. 1-13.

[4] Jason G Digalakis, Konstantinos G Margaritis "An Experimental Study of Benchmarking Functions for Genetic Algorithms" *Intern J Computer Math*, 2002 Vol. 79(4), pp 403-416.

[5] Kenneth A De Jong "an Analysis of the Behaviour of a Class of genetic adaptive systems" University of Michigan Press, Ann Arbor, 1975.