A Similarity-based Cellular Selection Mechanisim for Genetic algorithms to Solve Assignment Problems

Hossein Rajabalipour Cheshmehgaz
Universiti Teknologi Malaysis, UTM, Skudai JB, Malaysia
hr.pour@yahoo.com

Habibollah Bin Haron
Universiti Teknologi Malaysis, UTM, Skudai JB, Malaysia
habib@utm.my

Mohammad Reza Meybodi
Amirkabir University of Technology, Iran
meybodi@ce.aut.ac.ir

Abstract - In this paper, we illustrate a cellular structure mixed with Genetic Algorithms for solving assignment problems which have more than one feasible or optimum solution. Considering similarity among individuals in population, we use two dimensions Cellular Automata in order to place the individuals onto its cells to make the locality and neighborhood on Hamming distance basis. This new structure and using Genetic Algorithm on it, as 2D Cellular Automata Hamming GA, introduces locality for genetic selection and local knowledge for their selection process on cells of 2D Cellular Automata. The cellular selection based on the structure can ensure maintaining population diversity and fast convergence in the genetic search and improve the convergence performance during the genetic search.

Keywords-Genetic Algorithms; Assignment Problems; Cellular Automata; Optimization; NP-hard Multi Solutions Problems

I. INTRODUCTION

Genetic Algorithms (GAs) are used to solve difficult optimization problem like many NP-Hard Problems [1] in scientific and engineering areas. Resources distribution [2], management [3] and Assignment: Channel Assignment Problem [4], Graph Coloring Problem [5]; Multi-objective Optimization of Systems: 0/1 Multiple Knapsack Problem [6], Assembly Line Balancing Problem [7]; and etc are considered to solve by GA. But the main disadvantages of genetic algorithms are the disruption of good sub-solutions by crossover and mutation operations and undesired population diversity loss by selection operations, which constantly decreases the variety of its specimens. Population diversity should be preserved to prevent degeneration while maintaining the general trend of the evolution and some sort of selective pressure. Crossover operations are applied to produce new individuals from parents on the fitness which has been noticed by selection operation and Random mutations are applied to every new solution proposal in an attempt to slow down degeneration and introduce new characteristics to the population.

Several mechanisms have been integrated with genetic algorithms in various ways to preserve the diversity of species. Reference [8], Matousek and Nolle have presented a binary GA with a modified mutation operator, which is based on the well-known Hill Climbing Algorithm (HCA) and the selection operator preserves the best individual from the GA population during the selection process while maintaining the positive characteristics of the standard tournament selection. Chen and Wang [9] have presented a selection method combining roulette selection with tournament selection is presented to reinforce the local search ability. In [10, 11, and 12], some modified selection methods are proposed to increase the gain of resources, reliability and diversity; and decrease the uncertainty in selection process.

Locality, in selection, has been considered in some researches as one aim to increase the speed of finding a solution (feasible or optimum) [1]. In [14], the researcher has introduced the cellular automata as model, to realize the locality and neighborhood in the population structure. Based on the structure of cellular automata, the selection of individuals is controlled to avoid the fast population diversity loss during the genetic search.

Mostly, the problems which are mentioned in the beginning of this section have more than one optimum/feasible solution. Thin selection (built only on fitness basis) would disrupt the good solutions (different good solutions) and produce not better more offspring for new population. Therefore, considering the detail of the specific problem and then using thick selection (built on more detail basis), can improve the convergence performance during the genetic search.

In this paper, we introduce the property of multi feasible/optimum solution problems and cellular automata to realize the locality and neighborhood in the population structure, which is done by using a computational cellular model and mapping individuals based on hamming distances between them. The selection mechanism of individuals (in GA) is controlled by this model to avoid the fast population diversity loss during the genetic search, and then by using the similarity-based cellular selection, 2D Cellular Automata Hamming GA (2DCAHGA) is defined. We apply new idea on some well-known NP-hard Problem like graph coloring, Channel Assignment Problems and the outcome shows as simulation results.

II. GENETIC ALGORITHMS AND MULTI SOLUTION ASSIGNMENT PROBLEMS

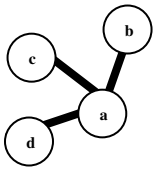
A. *Variety of solutions*

Genetic algorithms (GAs) solve complex optimization problems by manipulating a population of solutions by genetic operators like selection, crossover (recombination) and mutation [15]. For manipulating solutions by means of genetic operators, they have to be encoded in form of so-called individual (or chromosomes) each of which consists of a sequence of genes. For example, a simple graph coloring (NP-hard) problem is shown in figure 1, with 3 colors, there is a string of bits as matrix C that illustrates colors assigned to nodes of graph (in each row of matrix C). In this solution (not exactly correct), color 1 is assigned to nodes "a" and "d", color 2 is assigned to node "c"; color 3 is assigned to node "b".

Initially, GA starts with random generated solutions and makes the some individuals as initial population. Each individual would have a value calculated by a fitness function according some measures that show how much solution is closed to optimum/feasible solution. For instance (see figure 1) the value of fitness of solution can be calculated by penalty mechanism. Each wrong with the color assignment for each node would have one penalty. In this case, total of penalty is 1 (color 1 is assigned to adjacent nodes "a" and "d").

The general idea of a typical GA is best explained by the following scheme:

- Step 1: t= 0; *(Start with an initial time)*
- Step 2: init population P (t); *(Construct an initial population of individuals)*
- Step 3: evaluate P (t); *(Evaluate fitness of all Individuals of initial population)*
- Step 4: t: = t + 1; *(Increase the time counter)*
- Step 5: P_:= select parents P (t) *(Select a sub-population for offspring production on their fitness basis)*
- Step 6: crossover P_ (t); *(Do crossover the "genes" of selected parents)*
- Step 7: mutate P_ (t); *(Perturb the mated population stochastically)*
- Step 8: evaluate P_ (t); *(Evaluate its new fitness)*
- Step 9: P: = survive P, P_ (t); *(Select the survivors from actual fitness)*
- Step 10: while not do steps 4 through 9 *(Test for termination criterion (time, fitness, etc.))*
- Step 11: end GA *(Terminate the algorithm)*



Graph with 4 nodes

Colors 1 2 3	
a	1 0 0
b	0 0 1
c	0 1 0
d	1 0 0

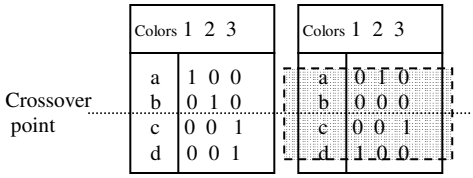
Color Assignment Matrix C
By 3 colors
(not exactly correct solution)

Figure 1. A Graph coloring Problem with 3 colors.

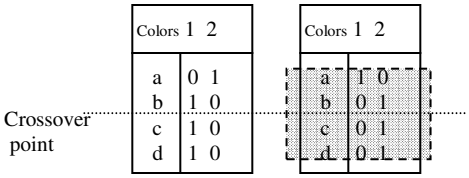
Considering graph coloring problem, the most important thing in such this case is having more than one feasible (or optimum) solution for the problem. Figure 2.a shows two different feasible solutions for previous case. However there are some better solutions with minimum number of colors as optimum solutions that are shown in figure 2.b.

Unfortunately, thin selection in GAs without notice to similarity among individuals may cause population diversity and low convergence in the genetic search. Both pair solutions in figures 2 have penalty with 0 and are best solutions for their own problem and objectives (with feasible solutions with 3-colors / optimum solutions with 2-colors), but if GA try to select these and do crossover on them (step 5 mentioned before), the result would not be better and the parents can not recreate offspring better than themselves.

Figure 3 shows offspring after crossover operation. As it is clear, the penalty for two new solutions is 1 (figure 3.a) and for others (figure 3.b) is 2. This retrogression has resulted from bad selection of couple to create offspring.

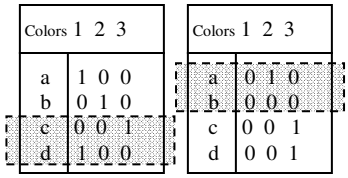


a. Color Assignment Matrix C as two feasible solutions with 3 colors

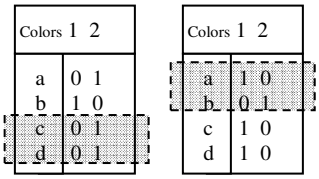


b. Color Assignment Matrix C as two optimum solutions with 2

Figure 2. Two feasible/optimum solutions for problem in figure 1.



a. Color Assignment Matrix C as offspring (after crossover) with 3 colors



b. Color Assignment Matrix C as offspring (after crossover) with 2 colors

Figure 3. Two feasible/optimum offspring after do crossover on individuals in figure 2.

B. Hamming Distance between Binary Solutions

Some researcher tried to modify the mechanism of selection in order to improve the GA [2, 3, 4 and 5]. In [14] Hamming distance is used for grading the difference between binary strings that have an equal number of bits. They consider a population with m individuals, and each individual include a binary string of length l. For every pair of binary strings $v_i = (b_1^i, \ldots, b_l^i)$ and $v_j = (b_1^j, \ldots, b_l^j)$, where b is equal to 0 or 1, the Hamming distance is defined by:

$$H(v_i, v_j) = \sum_{k=1}^{l} b_k^i \oplus b_k^j \qquad (1)$$

Where $\oplus$ satisfies a binary ‘‘exc1usive-or” operation.

To make offspring, GA selection operation selects first parent according the fitness as best (by roulette wheel selection and tournament selection techniques [15]), but to select second parent, in addition to fitness, notices the amount of hamming distance between new one and previous selected parent. Then, to make a decision to select, Modified new selections has two mechanisms: Selection-1 operation is considered for first parent and Selection-2 for second parent. As it is following, each selection has own mechanism but second one depend to first parent. So the value of second parent to select can modify as followed:

$$Value - for - SecondParent = f(a,b) \qquad (2)$$

a: Fitness of second parent.
b: Amount of hamming distance between first parent (selected parent) and second parent as candidate (selecting parent)

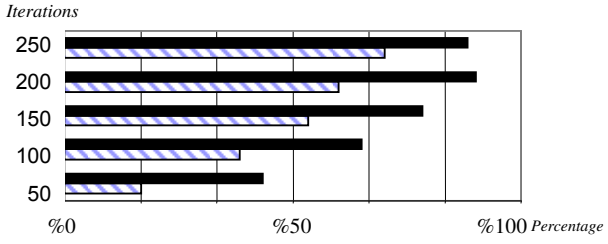
f is an incremental function depended on problem. For example it can be exponent function. The new GA called Hamming GA (HGA) can be changing the GA as followed in steps 5:

Step 5: // select a couple of individual for offspring production on their fitness basis for first one and on *Value-for-SecondParent* value basis for second.

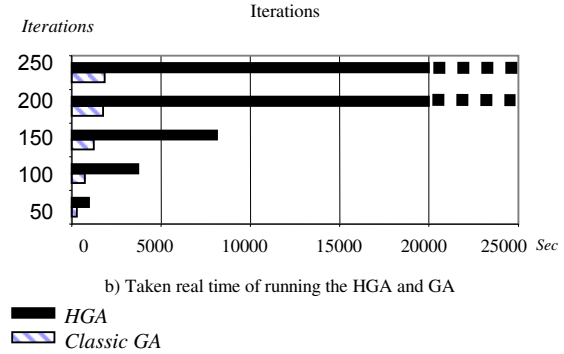
C. Comparing HGA and Classic GA

To comparing, HGA is applied on the well-known NP-hard Problem in cellular mobile networks, Channel Assignment Problem (CAP) [4] that has Multi feasible/optimum solution with three constraints. All three constraints are considered for the channel assignments: the co-channel constraint, the adjacent channel constraint and the co-site channel constraint. The goal of CAP is the assignment of the channel frequencies which satisfied these constraints with the lower bound number of channels (as feasible/optimum solution).

For instance, The CAP is considerate with 16 stations of BTS, the number of channels is 48 and the population size of GA is 100. The results are gathered and have been compared as is shown in figure 4. In figure 4.a, for iterations, there are 100 runs of each algorithm with the same parameters (crossover and mutation rate =90% and 5%, population size = 100, crossover type= vertical – see figure 2 and 3). The percentage axis in diagram shows number of runs that the algorithms have reached an optimum solution. Improvement of HGA in percentage of reaching an optimum solution in different iterations is sensible as compared to Classic GA (GA with no notice about Hamming Distance among individual). HGA could be mostly successful than classic GA to have a high chance to approach an optimum solution, especially in low iterations.



a) Percentage of reaching an optimum solution in different Iterations



b) Taken real time of running the HGA and GA

Figure 4. Comparing HGS with GA for CA Problem (Percentage of success to reach an optimum solution / Real time).

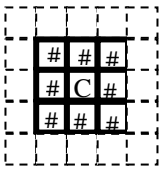


Figure 5. A part of 2DCA – C cell and its neighbors with mark #

But most important disadvantage is real time taken by the HGA. Both of algorithms run on Pentium 4 CPU 2.6GHz computer with 1GB RAM. As it is shown in figure 4.b, unfortunately, the real time would be high (for 250 iterations it takes more than 6 hours).

In next section, we apply 2-Dimensional Cellular Automat to reduce the real time of running HGA and it keeps the improvement as well.

III. 2D CELLULAR AUTOMATA HAMMING GENETIC ALGORITHM (2DCAHGA)

A. Cellular Automat: a computational Cellular Model

Cellular Automata (CA) are dynamical systems discrete in space, time and state variables and characterized by possession of exclusively local mechanisms of interaction. They constitute good models for the study of nonlinear complex systems. The popularity of cellular automata stems from their simplicity and transparency in definition; being discrete in all respects they are well-suited for computer experiments. But in spite of the simplicity in definition, the set of cellular automata contains many rules with very complicated behavior.

Formally, a CA is represented by the 4-tuple $\{Z, S, A, f\}$ where:

- Z is the finite or infinite lattice (all cells)
- S is a finite set of cell states or values (for this article, they are individuals and their Fitness Value)
- A is the finite neighborhood
- f is the local transition function defined by the transition table or the rule (for this article are selection, crossover and mutation operation)

The lattice is a finite or infinite discrete regular grid of cells on a finite number of dimensions. Each cell is defined by its discrete position (an integer number for each dimension) and by its discrete value. Time is also discrete. The future state of a cell (time t + 1) is a function of the present state (time t) of a finite number of cells surrounding the observed cell called the neighborhood (see figure 5, a 2DCA).

B. Mapping a population of GA onto a 2DCA

Before developing a 2DCAHGA, the individuals of each population are mapped onto all the cells of the cellular automaton, based on the Hamming distances.

Let us consider a population consisting of m individuals, with binary string v. We randomly choose v_i (i=1 … Size of Population), to be the element C (figure 5), then the neighborhood of C is constructed as follows. The individuals in population are sorted in ascending order of their corresponding Hamming distances to v_i. Choose the first eight elements v_k as the neighborhood of C, with other elements scattered outside its neighborhood, and then the whole population have been mapped onto a cellular automaton. The closest neighbors of C element are marked by # in figure 5. They usually have smaller Hamming distance to C. This idea has introduced locality in GAs and global knowledge for their selection process. The selection based on this cellular automaton can ensure maintaining population diversity and fast convergence in the genetic search

C. Steps of 2DCAHGA

For new GA, selection operation individually does for individuals in each cell and it would have a limitation to select a mate from only cell's neighborhoods. Each cell in CA, independently do selection and select one neighbor on the selection rules as own local transition function. This operation can be done in concurrency way too as extra advantage (in some parallel computers which can do). Each cell according to own position on CA would have some candidates to select as mate. Figure 6 illustrates one selection phase for all cells and own candidates to select (for 5×5 2DCA).

The 2DCA based Hamming genetic algorithm can, in essence, be given as follows:

Phase 1 (Making initial population)

- *Step 1:* generate initial population.

Phase 2 (Mapping on 2CA)

- *Step 2:* independently select an individual j from the current population. Map the selected individual j onto a cellular automaton, as described in the above subsection (previous sub section).
- *Step 3:* randomly chose one filled cell of CA such that has at leas an empty neighbor, and then select individuals and map just onto empty its neighbors as described before (previous sub section).
- *Sept 4:* if all cells of CA are not full, go to step 3.

Phase 3 (Cellular genetic actions)

- *Step 5:* independently, but they can concurrency, each cell select one of its neighbors as a mate to product a new offspring on their fitness and selection rules (as cellular selection).
- *Step 6:* in each cell, do crossover on individual in cell and its selected neighbor, and then do Mutation then, the one of the results would be kept instead of previous individual in the cell.

Phase 4 (Checking to finish)

- *Step 7:* if the best solution is not reached go to step 2.

There are 3 most important phases to do for 2DCAHGA and at the end of cellular genetic phase there is new population on cells. To continue the algorithm in order to reach a better solution, the phases would be continued (phase 4).

IV. Simulation Results

The new GA, 2DCAHGA is applied for the same CAP has mentioned in section II. Whatever, 2DCAHGA is able to apply on many multi solution problems such 0/1 multi knapsack, graph coloring, assembly line balancing for manufacturing, placement, and etc. as well as CAP and the results are the same.

There is a noticeable improvement in the real time taken by running 2DCAGHA as compared to HGA and Classic GA in figure 7.b. but there is not exactly specific behavior in low iterations (50 to near 150 iterations) to demonstrate that new mechanism of the selection would be better than the mechanism (of HGA) in percentage of reaching an optimum solution (see figure 7.a). As it is obvious, new mechanism works well (about 90%) in high iterations of running (250 iterations) and the real time (about 1000 sec) too. Totally, the results show that the cellular model is capable of searching feasible/optimum solutions in reasonable time and iterations.

V. Concluduion

By using cellular structure, two dimensional cellular automata based hamming genetic algorithm (2DCAHGA) has been developed to maintain population diversity during the genetic search, which is essential for genetic algorithms to solve the difficult Multi feasible/optimum solution problems such as 0/1 multiple knapsack, channel assignment in cellular mobile network, graph coloring, and many NP-hard problems which have more than one feasible or optimum solution. The cellular automaton is introduced to realize the locality and neighborhood in the population structure. The similarity-based cellular selection of individuals is controlled based on the structure of cellular model. Applications of 2CAHGA in optimization problems have shown that 2CAHGA is capable of searching for the global optimum solution of difficult nonlinear optimization problems.

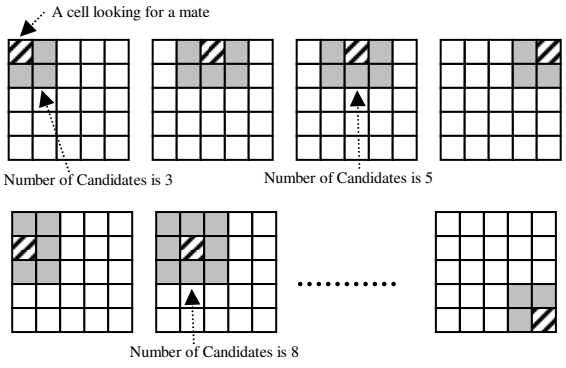


Figure 6. One selection phase for cells with their own candidates

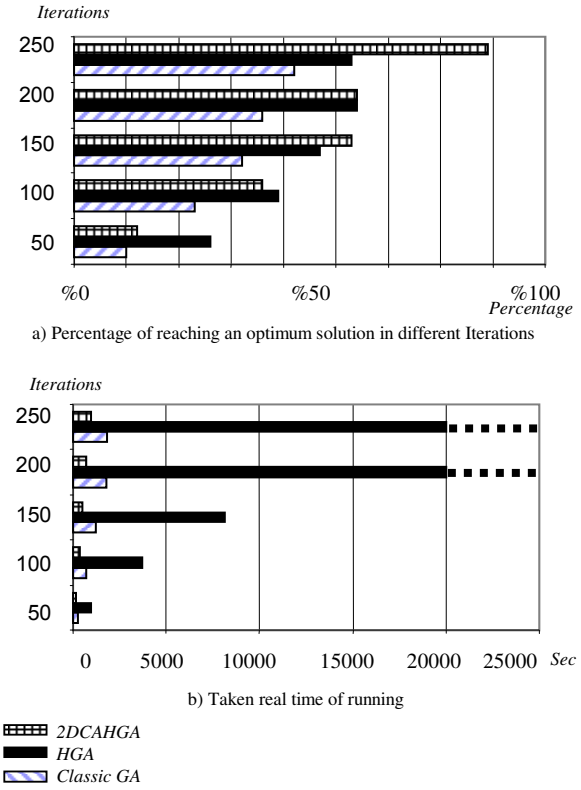


Figure 7. *Property of 2DCAHGA, HGS and Classic GA for CA Problem (Percentage of success to reach an optimum solution / Real time).*

Acknowledgment

This work was partially supported by a research student grant from the Universiti Teknologi Malausia, Skudai JB, Malaysia, and Project Vote: 79319, 2008.

References

[1] G. J. Woeginger, "Exact Algorithms for NP-Hard Problems: A Survey," Lecture notes in computer Science, Springer-Verlag, Germany, vol. 2570, 2003, pp. 185-207.

[2] S. Wang, and C. Ni, "Application of Projection Pursuit Dynamic Cluster Model in Regional Partition of Water Resources in China," Water Resources Management, vol. 22, 2008, pp. 1421-1429.

[3] L. C. Chang, "Guiding rational reservoir flood operation using penalty-type genetic algorithm," Hydrology, vol. 354, 2008, pp. 65-74.

[4] L.M. San Jose-Revuelta, "A new adaptive genetic algorithm for fixed channel assignment," Information Sciences, vol. 177,2007 ,pp. 2655-2678.

[5] J. Q. Yu, and S. N. Yu, "A novel parallel genetic algorithm for the graph coloring problem in VLSI channel routing," Proc. 3rd International Conference on Natural Computation, China, vol. 4, 2007, pp. 101-105.

[6] Y. Yoon, Y. Kim, and B. R. Moon, "An evolutionary Lagrangian method for the 0/1 multiple knapsack problems," Proc. Genetic and Evolutionary Computation Conference, South Korea, vol. 1, 2005, pp. 629-635.

[7] S. O. Tasan and S. Tunali, “A review of the current applications of genetic algorithms in assembly line balancing,” Intelligent Manufacturing, vol. 19, 2008, pp. 49-69.

[8] R. Matousek and L. Nolle, “GAHC: Improved Genetic Algorithm,” Proc. World Congress on Engineering and Computer Science, USA, 2007, pp. 915-920.

[9] Z. Q. Chen, R. L. Wang, and K. Okazaki, “An Efficient Genetic Algorithm Based Approach for the Minimum Graph Bisection Problem,” Computer Science and Network Security, vol.8, 2008.

[10] O. A. Jadaan, D. Rajamani, and C. R. Rao, “Improved Selection Operation for GA,” Theoretical and Applied Information Technology, vol. 4, 2008, pp. 269-277.

[11] R. Kumar, K. Izui, and Y. Masataka, “Multilevel Redundancy Allocation Optimization Using Hierarchical Genetic Algorithm,” IEEE Transactions on Reliability, vol. 57, 2008, pp. 650-661.

[12] F. Jaimes-Romero, and D. Munoz-Rodriguez, “Evolutionary searching in cellular radio systems planning,” European Transactions on Telecommunications, vol. 10, 1999, pp. 85-96.

[13] L. Di Gaspero and A. Schaerf, “Multi-neighbourhood local search with application to course timetabling,” Proc. 4th International Conference on Practice and Theory of Automated Timetabling, Belgium, vol. 2740, 2002, pp. 263-287.

[14] Y. J. Cao and H. Q. Wu, “A Cellular Automata Based Genetic Algorithm and its Application in Mechanic Design Optimization,” International Conference on CONTROL, 1998.

[15] Z. Ming and S. Shudong, “Theory of Genetic Algorithm and Its Application,” National Defense Industry publishing Company, Beijing; 1999, pp. 125-127.