Genetic Algorithm to Solve Sliding Tile 8-Puzzle Problem

Ruqaya Zedan Sha'ban
Computer unit.
College of Medicine
University of Mosul

Isra Natheer Alkallak
Department of Basic sciences
College of Nursing
University of Mosul

Mowada Mohamad Sulaiman
Department of chemistry
College of Sciences
University of Mosul

Received 09 / 06 / 2009 **Accepted** 02 / 11 / 2009

الخلاصة

تطرق البحث إلى دراسة مسألة اللغز-٨ والتي تعد من مسائل الذكاء الاصطناعي مع مفهوم الخوارزمية الجينية، اعتمد البحث على أساسيات الخوارزمية الجينية مع قواعد حل مسألة اللغز-٨، ابتداءاً من الحالة الحالية في رقعة اللعبة ومن ثم الوصول إلى هدف حل المسألة، اعتمد في حل المسألة على حركة الأماكن غير المتطابقة (خارج الموقع) في الحالة الحالية لرقعة اللعبة مقارنة مع هدف حل اللعبة دون حركة المكان الخالي (الفراغ). تم تحديد حجم المجتمع للخوارزمية الجينية المعدة من خلال المجموع الكلي لاحتمالات حركة الأماكن غير المتطابقة، في حين تم استخدام طريقة الترتيب في عمليتي التزاوج والتبادل. أثبتت نتائج البحث كفاءة الخوارزمية. أعدت برنامج حاسوبية بلغة ماتلاب لحل مسألة البحث.

ABSTRACT

The research tackled the classical problem in artificial intelligence as 8-puzzle problem with genetic algorithm. The research present the fundamental of genetic algorithm with sliding tile 8-puzzle problem. Starting from current state for state space search into a goal state by depending on the tile's move (tiles out of place) in the current and compare with the solution of the problem (goal), without blank's move. population size chose by the summation of probabilities misplaced tile's move (tiles out of place) in current state comparing with goal state. In this research, depended on the Crossover and mutation for ordered chromosomes method. The experimental in this research show that the algorithm is efficient. The source code is written in Matlab language.

1. Introduction

The 8 puzzle is a game invented by Sam Loyd in 1870s [12]. The 8-puzzle is a square tray in which are placed 8 square tiles. The remaining ninth square is uncovered. Each tile has a number on it. A tile that is adjacent to be the blank space can be slid into that space. A game consists of a starting position and a specified goal position. Rich [8].

The sliding tile puzzle is also called the n-puzzle, which features *n* tiles numbered from 1 to n and one blank tile in a square grid. The n-puzzle is known in various forms, the most famous being the 8-puzzle and 15-puzzle. A puzzle start with a jumbled arrangement of these tiles. A player can slide an adjacent tile into a position occupied by the blank tile. The goal of this game is to move the tiles so as to reach the goal state where all numbers are placed in an increasing order from left to right and from top to bottom Qian [6]. The objective of the 8-puzzle is to rearrange a given initial configuration of eight squared tiles on a 3x3 board into a specified goal configuration by successively sliding tiles into the orthogonally adjacent empty square (the blank square). While it would seem easy to find any solution to this problem, we are only interested in obtaining optimal solutions with the fewest moves. There exist 9! possible tile permutations on a 3x3 board, and every second permutation is solvable, there is a total of 9!/2= 181440 solvable problem instances Reinefeld [7].

A genetic algorithm is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are a particular class of evolutionary algorithms (also known as evolutionary computation) that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover also called recombination.

Genetic algorithms are implemented as a computer simulation in which a population of abstract representations called chromosome or the genotype or the genome of candidate solutions called individuals, creatures, or phenotypes to an optimization problem evolves toward better solutions.

The evolution usually starts from a population of randomly generated individuals and happens in generations. In each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population based on their fitness, and modified recombined and possibly randomly mutated to form a new population. The new population is then used in the next iteration of the algorithm Mitchell [5].

Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population. If the algorithm has terminated due to a maximum number of generations, a satisfactory solution may or may not have been reached Qian [6], [10],[11].

The research aims to present the fundamental of heuristic genetic algorithm with sliding tile 8-puzzle problem. Starting from current state for state space search into a goal state by depending on the tile's move (tiles out of place) in the current and compare with the solution of the problem (goal), without blank's move not as classical methods

1-2 Methodology of Genetic Algorithm

A genetic algorithm operates through a simple cycle of stages Konar [3].

i. Creation of a "population" of strings.
ii. Evaluation of each string.
iii. Selection of best strings.
iv. Genetic manipulation to create new population of string.

Figure (1) illustrated the cycle of a genetic algorithm Konar [3].

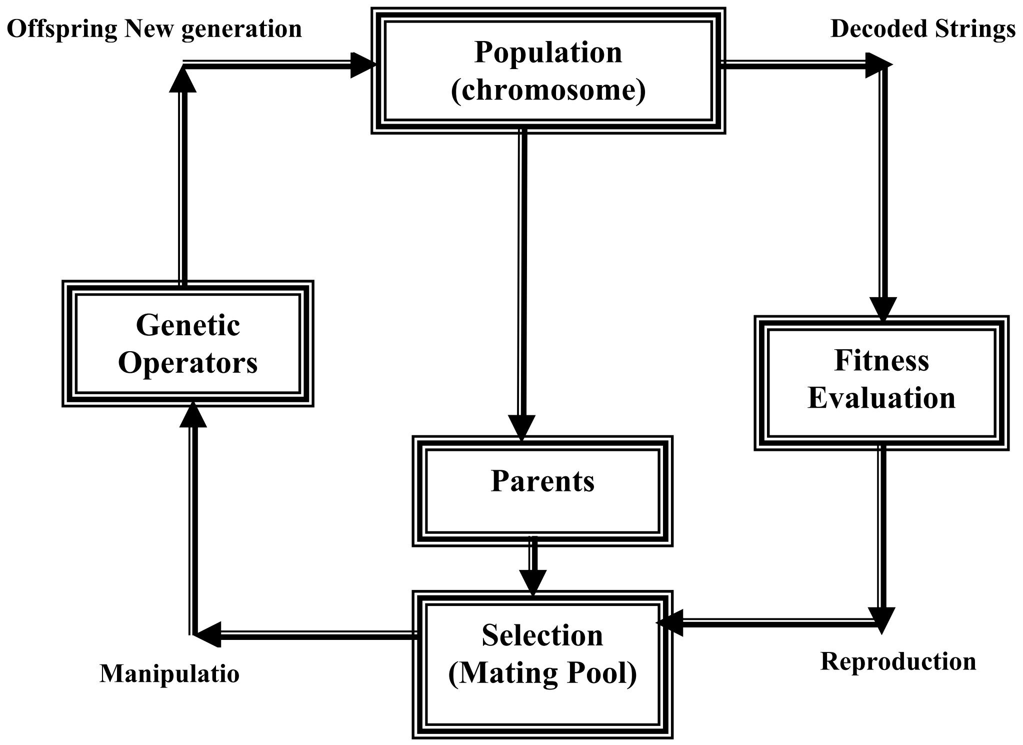
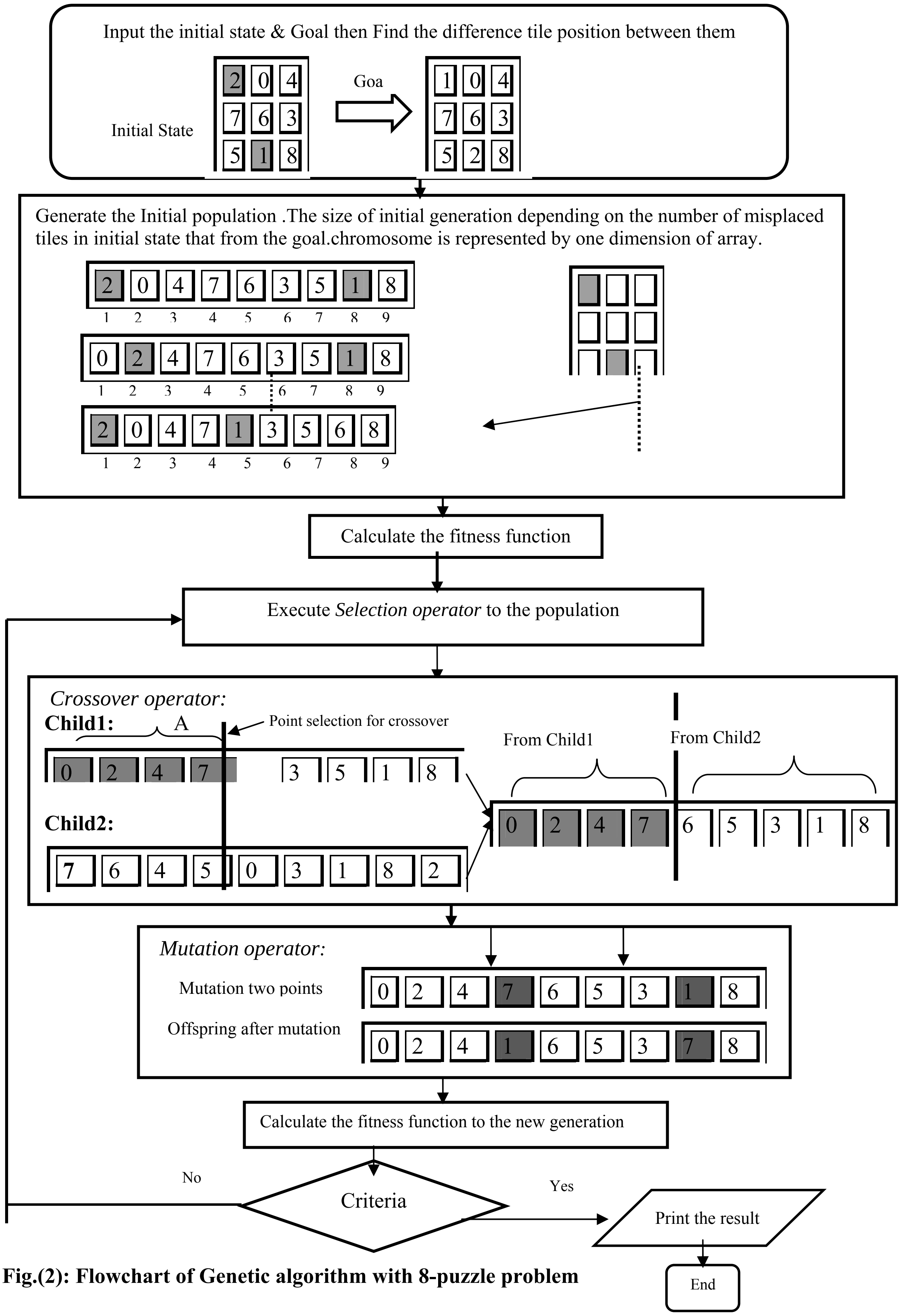


Figure (1): The cycle of a genetic algorithm

2. Flowchart for proposed genetic algorithm with 8-puzzle problem

Below figure **(2)** Flowchart for proposed genetic algorithm with 8-puzzle problem



Input the initial state & Goal then Find the difference tile position between them
Initial State
Goa
Generate the Initial population .The size of initial generation depending on the number of misplaced tiles in initial state that from the goal.chromosome is represented by one dimension of array.
Calculate the fitness function
Execute *Selection operator* to the population
Crossover operator:
Child1:
A
Point selection for crossover
Child2:
From Child1
From Child2
Mutation operator:
Mutation two points
Offspring after mutation
Calculate the fitness function to the new generation
No
Criteria
Yes
Print the result
End

Fig.(2): Flowchart of Genetic algorithm with 8-puzzle problem

2-1 Create the Individuals and Population

Initially many individual solutions are randomly generated to form an initial population size depends on the nature of the problem, covering possible solutions (the search space) Mitchell [5].

In this research, the population size was equal the number of tiles out of place therefore, this is depended on the number of generations. The length of each Individuals varies from one to another with a lower bound of 2 and an upper bound of 24 in 8-puzzles. Also in this research, uses a 9 bits to represent the chromosome.

for example if tile in the index 3 and tile in the index 4 is out of place, then tile in the index 3 has 2 tile's moves (left, down) and tile in the index 4 has 3 tile's moves (right, up, down) therefore the population size must be 5 chromosome.

are shown next:

The problem :

2	0	4
7	6	3
5	1	8

The chromosome

2	0	4	7	6	3	5	1	8

2-2 Selection

During each successive process, a proportion of the existing population is selected generation. Individual solutions are selected through a fitness-based process. The fitness of each solution and preferentially select the best solutions Gen *et.al* [1].

In this research, used tournament selection method that is one of many methods of selection in genetic algorithms which runs a tournament among a few individuals chosen at random. from the population and selects the winner (the one with the best fitness) for crossover. Below the steps of tournament selection procedure see Mitchell [5] :

1- Choose two individuals at random from population.

2- Choose a random r between 0 and 1.

3- Choose k is parameter, for example 0.75

If $r < k$ then the fitter of the two individuals is selected to be parent.

Otherwise the less fit individual is selected.

4- The two are then returned to the original population and can be selected again.

```
[p1,p2] = select(p1,p2,geel);   % where p1,p2 two chromosome chooses at
                                % random from function called select and variable
                                % geel is the length of population
r=rand(1)                       % Choose a random r between 0 and 1.
If prob(p1) > prob(p2)          % Compare the probability of p1 & p2
    great=p1;
    small=p2;
Else
    great=p2;
    small=p1
End
K = 0.75                % Choose k is parameter, for example 0.75
    If r < k
        great;          % then the fitter of the two individuals is selected to be parent.
    Else
        small;          % Otherwise the less fit individual is selected.
 End
```

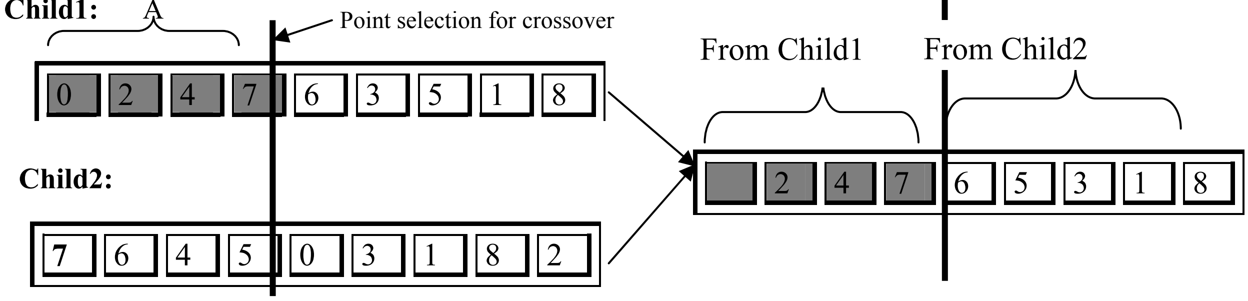
The two are then returned to the original population and can be selected again.

2-3 Crossover

The next step is to generate a second generation population of solutions from the genetic operators. Crossover (also called recombination). For each new solution to be produced, a pair of "parent" solutions is selected from selected previously. By producing a "offspring" solution. The process continues until a new population of size is generated. The Crossover operator used to generate the next population and have many crossover techniques Goldberg [2].

In this research, in Crossover operation used the ordered chromosomes method. A crossover point is selected on the parents. Since the chromosome is an ordered list, a direct swap would introduce duplicates and remove necessary candidates from the list. Instead, the chromosome up to the crossover point is retained for each parent. The information after the crossover point is ordered as it is ordered in the other parent as represented by:

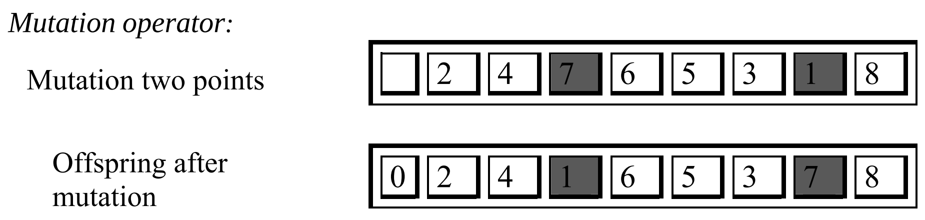
Crossover operator:



2-4 Mutation

The mutation is a genetic operator used to maintain genetic diversity from one generation of a population of chromosome to the next. A common method of implementing the mutation operator involve generating a random variable for each bit in a sequence. The purpose of mutation in genetic algorithms is to allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other Koza [4].

In this research, used the order changing method for mutation operator. As represented by :



2-5 Fitness Function

The fitness function depends on the nature of the problem. It computed for every chromosome by depending on the objective function the chromosome Schmidt *et.al* [9].

We express in this research, the proposed fitness function is calculated by:

$$F = \sum_{i=1}^{9} n * i$$

n : represent the tile's value

i : index of tile

2-6 Termination

This generational process is repeated until a termination condition has been terminating. many runs were performed, stop criterion was used like classical measures to accept (reach) the solution and depends on the nature of the problem Goldberg [2].

In this research, classical measures used to reach the goal depend on the number of generation.

3. The Steps of Proposed Heuristic Genetic Algorithm

Below the illustrated steps of the proposed heuristic genetic algorithm for solving the 8-puzzle problem :

Step 1: Create an initial state matrix of the problem as 3x3 grid filled numbers 1-8 and a blank, the length of chromosome represented by:

١	٤	٣
٧	٦	0
٥	٨	٢

$\Rightarrow$ [1 4 3 7 6 0 5 8 2]

The number 0 is used to represent the blank square

Similarly, the goal state of the problem could be represented :

١	٤	0
٧	٦	٣
٥	٨	٢

$\Rightarrow$ [1 4 0 ٧ ٦ ٣ ٥ ٨ ٢]

Step 2: Create the initial generation (population) randomly, population size is the number of misplaced tiles (tiles out of place) in initial state, As below in the table (1). Represented the number and direction of tile's move

Index of tile	The number of tile's move	The direction of tile's move
١	٢	D,R
٢	٣	R,L,D
٣	٢	D,L
٤	٣	U,D,R
٥	٤	U,D,R,L
٦	٣	U,D,L
٧	٢	U,R
٨	٣	U,R,L
٩	٢	U,L

Where R: Right, L: Left, U: Up, D: Down

The initial state is some arbitrary arrangement of the tiles. An initial state and goal state are shown next :

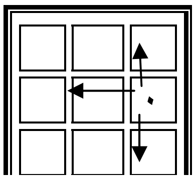
1	٤	٣
٧	٦	٠
٥	٨	٢

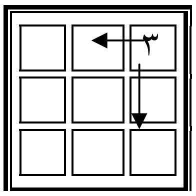
Initial state

١	٤	٠
٧	٦	٣
٥	٨	٢

Goal state

For above example, in the initial state the population size is 5 depend on the tile's moves, when the tile in the index 3 has 2 tile's moves, and the tile in the index 6 has 3 tile's moves are shown next :





Step 3: let *it* =1 ; the *it* is a variable of initialize the iteration number of generation

Step 4: Compute fitness value (fitness function) for each chromosome in the generation for proposed heuristic genetic algorithm with 8-puzzle problem is given as:

$$F = \sum_{i=1}^{9} n * i \qquad n: \text{ represent the tile's value}$$

$$i: \text{ index of tile}$$

and Compute the probability of fitness function by :

$$Prob_i = F_i / \sum_{i=1}^{9} n * i$$

Step 5: Compare the fitness function for each chromosome with fitness function of the goal, and also compare the tile's value of chromosome with tile's value of the goal. If the two conditions are satisfied, then record the generation's index and chromosome index. Else go to the next chromosome.

Step 6: The next generation are produced by executing Selection, Crossover and Mutation operations respectively.

(a): Tournament Selection operator to select the fitter parents.

(b): Execute the crossover operator consists of combining parts of individuals to create new individuals as following:

```
Begin
Initialize
Let Probability select $pc$ = 0.7 ; where ($0.5 < pc < 0.8$)
Let $point$ = randint (length of chromosome); determine the
      crossover point
Let $ra$ = size of population
Let $k$ =1      ;   k is a counter
**While** $k <= ra$
- Let p1 , p2 pairs of individuals choose d randomly from the
      population.
- Let r a random number ; where ($0 < r < 1$).
**IF** $r > pc$   (if true then crossover operator is executed)

    The first part of *chromosom* about (child 1) is the same as of
    *chromosom* in p1 before crossover p*oint*
    Also the first part of *chromosom* about (child 2) is the same as
    of *chromosom* in p2 before crossover p*oint*
End
```

```
    -Execute the crossover for ordered chromosomes by *Order procedure* to
    create the remained chromosome of *child1* and *child2* after crossover point.
    Else
        point = length of chromosome
        the all chromosome of (chile1) and (child2) is the same of chromosome as in
        p1, p2
    End
    k = k+2    ;    increase the counter by 2
End
```

(c): Execute Mutation operator

Used order changing method as two numbers are selected and exchanged.

```
Begin
  Let Probability mutation bit selected (mutation rate) as $pm = 0.011$; where $(0.01 < pm < 0.5)$
  {The Probability mutation bit selected equal the inverse of size of population multiply length of chromosome as $pm = 1/ra * length\ of\ chromosome$ }
  Let  $K = 1$
  While  $k < ra$
    Let $rr$ a random number ; where  $(0 < rr < 1)$.
    IF  $rr < pm$  (if true then mutation operator is executed)
        point1 = randint (length of chromosome)
        point2 = randint (length of chromosome)
        swap values in two columns positions point1 and point 2 in child(k)
    Else
        point = length of chromosome
        the chromosome in child (k) stay same as not change.
    End
    k = k+1  ; increment the counter
  End
Step 7:  it=it+1 increment the iteration number of generation.
         IF it equal the number of generation THEN step8
         Else Return to step 4
Step 8:  Stop
```

3-1 Running the Proposed Heuristic Genetic Algorithm for Solving the 8-Puzzle

Problem :
Below two runs of the program in this research.
Run 1:

١	٤	٣
٧	٦	
٥	٨	٢

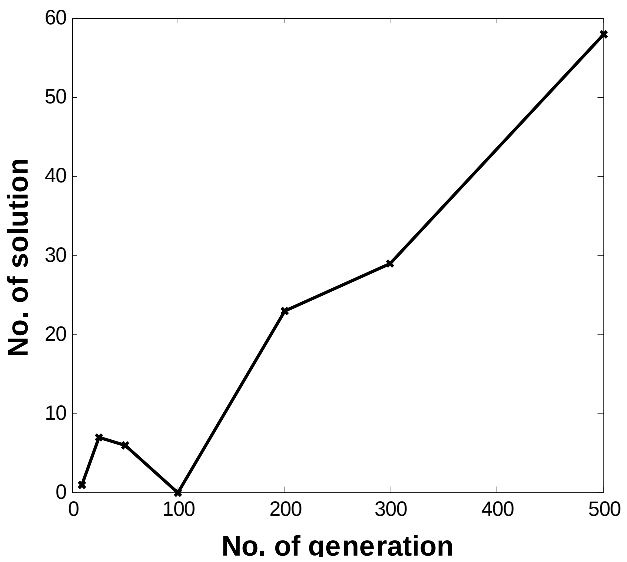
١	٤	
٧	٦	٣
٥	٨	٢

Initial state
Goal state

Number of different tiles is 2. therefore length of initial generation is 5. Below table (2) shows the execution of proposed algorithm and the figure (3) shows the number of solution with number of generation for run 1:

No. of generation	No. of solution	The generation that is found solution in it
10	1	1
25	7	5,8,16,20,22,23,25
50	6	1,12,16,19,46,49
100	No solution	-
200	23	1,2,13,17,18,24,30,31,48,51,63,72,74,107,109,113,133,144,145,160,162,168,200
300	29	19,20,27,39,41,49,80,89,91,111,113,116,119,120,123,134,143,145,149,155,163,180,203,205,216,245,250,256,282
500	58	48,49,57,63,66,72,74,90,97,98,101,120,138,142,146,147,160,163,164,165,168,182,183,215,217,224,227,254,258,260,268,271,272,277,283,288,298,299,302,325,327,340,346,355,387,389,405,406,413,416,420,424,425,427,445,454,459,490

Table(2): the execution of proposed algorithm for run 1



Figure(3): the number of solution with number of generation for run 1

Run 2:

١	٤	٣
٧		٦
٥	٨	٢

Initial state

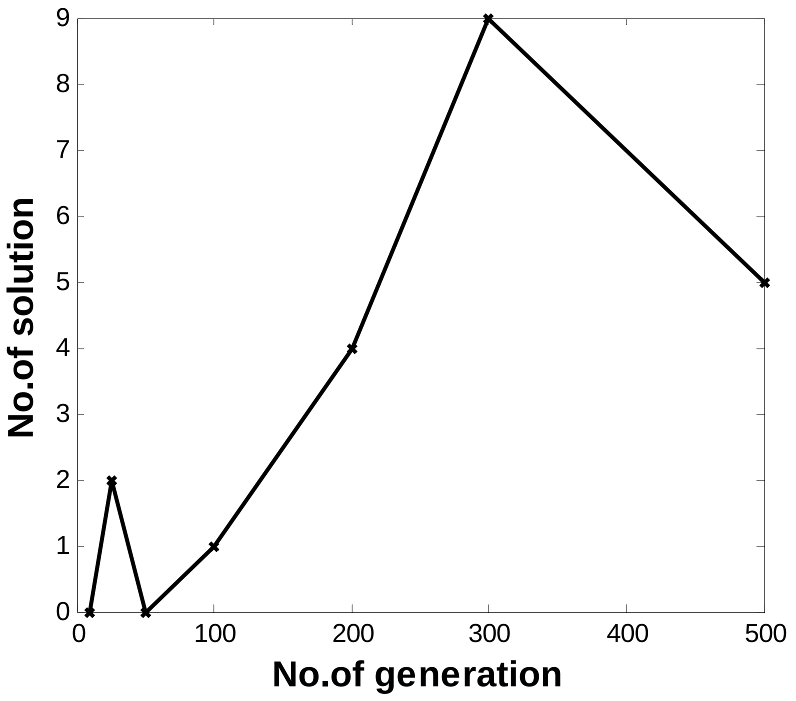
١	٤	
٧	٦	٣
٥	٨	٢

Goal state

Number of different tiles is 3. therefore, length of initial generation is 9. Below table (3) shows the execution of proposed algorithm and the figure (4) shows the number of solution with number of generation for run 2 :

No. of generation	No. of solution	The generation that is found solution in it
10	No solution	-
25	2	17,24
50	No solution	-
100	1	24
200	4	39,88,131,139
300	9	1,4,10,7,6,3,5,8,2
500	5	21,119,406,420,424

Table (3): the execution of proposed algorithm for run 2



Figure(4): the number of solution with number of generation for run 2

3-2 Conclusion

The sliding tile puzzle is a typical problem for modeling algorithms involving heuristics. The proposed heuristic genetic algorithm for solving the 8-puzzle is feasible plan, because heuristic rules can be used as fitness function to evaluate the individuals that have evolved in each stage. It founds the feasible solution for a state space search that will enable us to find a series of moves that transform a start puzzle into a goal puzzle. The proposed heuristic genetic algorithm has the ability to find a global solution in a large space. The algorithm performs significantly bettor than the traditional search methods. The dynamic of population size was equal the number of tiles out of place therefore, this is depended on the number of generations in evolutionary computed have been reached to the goal and reduce the region of the space considered.

References

1) Gen, M., and K. Ida. (2000), spanning tree-based genetic algorithm for bicriteria fixed charge transportation problem, Journal of Japan society of fuzzy theory and systems.

2) Goldberg, D. E. (1989), genetic algorithms in search, optimization and machine learning, Addison, Wesley.

3) Konar, A., 2000, Artificial Intelligence and Soft Computing Behavioral and Cognitive Modeling of the Human Brain, CRC press, Inc., Boca Raton London New York Washington, D.C., pp.450.

4) Koza, J. R. (1992), genetic programming, MIT press, Cambridge, MA.

5) Mitchell, M., 1998, An Introduction to Genetic Algorithms, MIT press, London.

6) Qian, T., 1995, Using Genetic Algorithm to Solve Sliding Tile Puzzles, Cognitive Science Program, Oswego, USA.

7) Reinefeld, A. 2006, Complete Solution of the Eight-Puzzle and the Benefit of Node Ordering in IDA*, Paderborn Center for Parallel Computing, Germany.

8) Rich, E. 1988, Artificial Intelligence, McGraw-Hill, Inc., Eight Edit, Singapore.

9) Schmidt, M. and Stidsen, T. (1997), genetic algorithms, Neural networks and fuzzy logic, DAIMIIR.

10) http:// en.wikipedia.org /wiki /genetic algorithm

11) http://en.wikipedia.org/wiki/mutation_%28genetic_algorithm%29

12) http://en.wikipedia.org/wiki/Sliding_puzzle