2012 International Conference on Artificial Intelligence and Soft Computing
*Lecture Notes in Information Technology, Vol.12*

# Solving Sudoku with New Genetic Algorithm

Xiuqin Deng[1,a], Yongda Li [2,b] and Ruichu Cai[3,c]

[1] Faculty of Applied Mathematics, Guangdong University of Technology, Guangzhou 510006, China

[2] Tencent Technology Company Limited, Shenzhen 518057, China

[3] Faculty of Computer Science, Guangdong University of Technology, Guangzhou 510006, China

[a]xiuqindeng@163.com, [b]756602820@qq.com, [c]cairuichu@gmail.com

**Keywords:** Sudoku, Genetic algorithm, Meta-heuristics algorithm, Convergence.

**Abstract.** Sudoku is an interesting problem because it is a challenging logical puzzle that has recently become widely popular in around the world and internet. Sudoku can be regarded as a constraint satisfaction problem. In this paper, a new genetic algorithm is proposed. First, the selection operator, crossover operator and mutation operator of the genetic algorithm have effectively been improved according to features of Sudoku puzzle. Secondly, in order to avoid the problem of determining the probability of crossover and mutation, a “random” technology is applied in new genetic algorithm. The solving efficiencies of new genetic algorithm and other meta-heuristics algorithm are compared. A large number of simulation experiment results showed that new genetic algorithm has higher convergence speed and convergence ratio in solving Sudoku puzzles.

## 1. Introduction

Sudoku has been claimed to be very popular and even addictive because they are very challenging but have relatively simple rules [1]. According to Wikipedia [2] Sudoku is a Japanese logical game that has recently become hugely popular in Europe and North-America. However, the first Sudoku puzzle was published in a puzzle magazine in USA as early as 1979, after which it circled through Japan, where it became popular in 1986. In 2005, it became a worldwide phenomenon [3].

Sudoku puzzles are composed of a $9\times9$ grid, namely of 81 positions, which are then divided into nine $3\times3$ sub-grids. Once Sudoku puzzle is ready to play there are initially some static numbers (givens) that are not allowed to be changed or moved during a process of solving Sudoku puzzles, the solution of Sudoku puzzles is such that each row, column and $3\times3$ sub-grids contains each integer $\{1,2,3,4,5,6,7,8,9\}$ once and only once [2].The puzzle is presented so that in the beginning there are some static numbers (givens) that cannot be changed or moved. The number of givens does not determine the difficulty of the puzzle. Instead, there are 15 to 20 factors that are claimed to have an affect to the difficulty rating [1]. The givens can be symmetrical or nonsymmetrical. In the symmetric case, all the givens are located symmetrically with respect to the centre position.

The Sudoku problem seems to have increased popularity as a research problem recently. The Sudoku problem could be seen as a good test bench for algorithm design and representation, since it is generally known [4]. Because Sudoku is an NP-complete problem, various heuristics algorithm was used for solving Sudoku puzzles. Nicolau and Ryan [5] developed a system named GAuGE for Sudoku, which uses a position independent representation. Each phenotype variable is encoded as a genotype string along with an associated phenotype position to learn linear relationships between variables. Geem [6] have used harmony search (HS) algorithm to solve Sudoku puzzle. Their results showed that HS could successfully solve the easy Sudoku but it failed to find the global optimum for hard level with 26 given values. The HS model was instead entrapped in one of local optima with the penalty of 14 after 1,064 function evaluations. Perez and Marwala [7] have used many different methods: cultural genetic algorithm, repulsive particle swarm optimization, quantum simulated annealing, and genetic algorithm/simulated annealing hybrid (HGASA) for solving Sudoku. Their results showed that the HGASA method was the most efficient of them when solving Sudoku. Mantere and Koljonen [4,8,9] have tried various evolutionary algorithm (EA) methods , i.e., genetic algorithm (GA), cultural algorithm (CA), ant colony optimization (ACO) and genetic algorithm/ant colony optimization hybrid (GA/ACO) for solving Sudoku. Their results revealed that GA/ACO was more efficient than any of the other three methods and the analysis of the hybrid method and its parameter settings helped improve the GA and CA methods, too, and their results also improved by 14.2% and 19.4%, respectively.

In order to enhance the convergence speed for solving Sudoku puzzles based on genetic algorithm, Li and Deng [10] have improved the various important operators of the genetic algorithm in a bold way, so that the solved Sudoku puzzles had higher reliability, better stability and quicker convergence speed. But the crossover probability and mutation probability of improved genetic algorithm (IGA) [10] are constant during the optimization, thus to affect the stability and practicability of the algorithm and to restrict the convergence speed to a certain degree. In view of the features for solving Sudoku puzzles, this paper effectively improves the selection operator, crossover operator and mutation operator. In order to avoid the problem of determining the probability of crossover and mutation,  a “random” technology is used in this paper (see section 3.2、3.3 for more detail).

## 2. Fitness function and Sudoku encoding

The key point for solving Sudoku puzzles by successfully using all kinds of meta-heuristics algorithm is how to design a fitness function and how to encode a solution（chromosome）. Although the sum of each row, each column, or each block equals 45, it does not guarantee that the numbers 1 through 9 are used exactly once. Figure 1 show the solution of Sudoku puzzle of **Fig**.1 (easy level) of Geem [6] and Figure 2 is a local optimal solution, however both of penalty value are 0.So fitness function of HS algorithm [6] is error. In this paper, the fitness function is defined as the sum of repeated digits in each row, each column. Obviously a solution with penalty of 0 must be the global optimal solution. Sudoku coding in this paper is similar as Mantere & Koljonen[8].

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 5 | 4 | 3 | 1 | 6 | 8 | 9 | 7 |
| 7 | 6 | 3 | 9 | 8 | 5 | 1 | 2 | 4 |
| 1 | 9 | 8 | 4 | 2 | 7 | 6 | 5 | 3 |
| 9 | 8 | 1 | 7 | 5 | 3 | 2 | 4 | 6 |
| 6 | 3 | 2 | 8 | 4 | 9 | 7 | 1 | 5 |
| 5 | 4 | 7 | 2 | 6 | 1 | 9 | 3 | 8 |
| 4 | 7 | 5 | 6 | 9 | 2 | 3 | 8 | 1 |
| 3 | 1 | 9 | 5 | 7 | 8 | 4 | 6 | 2 |
| 8 | 2 | 6 | 1 | 3 | 4 | 5 | 7 | 9 |

Fig.1 The global optima with penalty of 0

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 5 | 4 | 3 | 1 | 6 | 8 | 9 | 7 |
| 7 | 6 | 3 | 9 | 8 | 5 | 1 | 2 | 4 |
| 1 | 9 | 8 | 4 | 2 | 7 | 6 | 5 | 3 |
| 9 | 8 | 1 | 8 | 4 | 3 | 2 | 4 | 6 |
| 6 | 3 | 2 | 7 | 5 | 9 | 7 | 1 | 5 |
| 5 | 4 | 7 | 2 | 6 | 1 | 9 | 3 | 8 |
| 4 | 7 | 5 | 6 | 9 | 2 | 3 | 8 | 1 |
| 3 | 1 | 9 | 5 | 7 | 8 | 4 | 6 | 2 |
| 8 | 2 | 6 | 1 | 3 | 4 | 5 | 7 | 9 |

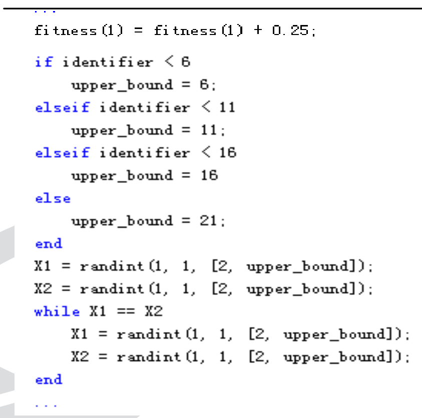
Fig.2 A local optima with penalty of 0

## 3. Genetic operator of new algorithm

### 3.1 Selection operator

Any population will generate 20 chromosomes, then there are 41 chromosomes for us to choose. As we know, each chromosome has its corresponding fitness function value $w_i$, $i \in \{1,2,3,...,21\}$. All of best 21 chromosomes are then encoded with numerical data linearly ordered by magnitude against the value of $w_i$ in which the chromosome with smallest value is encoded as 1 while with largest value as 21. The 21 chromosomes with encoded numbers in the range of 1-21 are selected to generate the next generation of population. Then we will choose 20 pairs of chromosomes from this generation with the principle show as *Table*1.

Table 1 Selection operator of new algorithm



```
...
fitness(1) = fitness(1) + 0.25;
if identifier < 6
    upper_bound = 6;
elseif identifier < 11
    upper_bound = 11;
elseif identifier < 16
    upper_bound = 16
else
    upper_bound = 21;
end
X1 = randint(1, 1, [2, upper_bound]);
X2 = randint(1, 1, [2, upper_bound]);
while X1 == X2
    X1 = randint(1, 1, [2, upper_bound]);
    X2 = randint(1, 1, [2, upper_bound]);
end
...
```

Since the smaller fitness of population is the less of numbers more optimal than the optimal feasible solution of the current population, the more difficult to find the more optimal solution with crossover operator. *Table* 1 indicates the fitness of the optimal chromosome must be added with 0.25, such that the optimal chromosome after four times of iteration remains unchangeable, resulting in an increase of fitness of optimal chromosome started from the 5th iteration by 1, therefore, the near-optimum chromosome (with difference of fitness by 1 to the optimal chromosome) will replace the optimal chromosome, such that the evolutionary direction of population genes is changed in crossover operation. Not only such a selection strategy can ensure the optimal chromosome in population certainly appeared in the next generation, but also in the process of selecting 20 pairs of parental chromosomes, the 20 item of chromosomes will be divided into four levels of priority, each of being 2~5, 6~10, 11~15, 16~21, respectively (where $k = 2, 3, \cdots, 21$ is an encoded number of chromosome). Any two chromosomes having the same level of priority have the identical priority during selection, such that the similarity of selected chromosome would be weakened, thereby the probability of chromosome having more abundant gene for selection is reasonably enlarged. The selection operators of new algorithm not only ensure the principle "survival of the fittest" but also ensure that the optimal chromosome in population certainly appeared in the next generation. In the other side, adding 0.25 to the old best fitness value each generation might make more sense than adding one as Mantere & Koljonen [8] have done.

## 3.2 Crossover operator

In this paper, we takes a $3 \times 3$ sub-blocks as a crossover- point and both cross-point position and crossover probability are a random number. In order to enlarge the searching range the numbers of cross-points is no longer an assigned constant, but a random variable k. K sub-blocks are taken from $X_1$ chromosome arbitrarily, where k is any one of integers from 1 to 8, and the rest $(9-k)$ sub-blocks are taken from $X_2$ chromosome to form a new chromosome. In this way, the searching direction for each iteration searching process is more flexible and the searching scope is wider. The convergence process shown in Table 3 and 4 (see section 4.1) indicates that the crossover operator of new algorithm can find the more optimal solution more quickly.

In the other side, renewal of particle state in PSO is implemented using three aspects of information, current position, empirical position and neighboring empirical position of the particles to adjust its state [11]. This information exchange mode of PSO is tactfully applied in crossover operator of new algorithm. Since two selected chromosomes are different one another, thus their encoded numbers are different, either, where *max* represents for a larger number of encoded chromosome while *min* for a smaller number of encoded chromosome and best for optimal chromosome (i.e. chromosome with number 1), $\lambda_1$ and $\lambda_2$ are the numbers of sub-blocks taken from one chromosome which are a random natural number in the range of 1-8, $\otimes_{\lambda_1}$ is defined in crossover operation with coefficient $\lambda_1$, thus the crossover operator could be represented as follows:

$$x_i = ( best \otimes_{\lambda_1} \max ) \otimes_{\lambda_2} \min \quad \lambda_1 \neq \lambda_2 , i = 2 , 3 , \cdots , 21 \tag{1}$$

The renewal of chromosome is affected by dual effects of self-experience and population experience, thereby making the crossover operator highly directional.

## 3.3 Mutation operator

The mutation probability of new algorithm changes along with the fitness value of the optimal solution in the current population in order to maintain the diversity of population gene, which makes it possible for the crossover operates to search and find the solution different with the current population. The detailed mutation probability is shown in *Table* 2. From *Table*2, we can see that the mutation probability changes as below: When the fitness value of the current population's optimal solution is higher than 10, the population gene is rather rich (as the convergence speed is rather quick when the fitness value is lower than or equal to 10, which indicates the rather great probability for the cross operators to search and find new chromosomes), with only one-time mutation for each new chromosome. When the fitness value of the optimal solution in the current population is 7, 8, 9 and 10, the mutation times for each new chromosome is a random integer of 1-5 generated to solve the problem that it is impossible for the genes to generate new chromosomes when they tend to be alike along with the convergence process. When the fitness value of the optimal solution in the current population is lower than or equal to 6, the genes tend to be alike still more, so it is necessary to raise the mutation probability. But the experiment results indicate that too great mutation probability may bring about vibration (which can generate solutions different with those in the current population, but the fitness value is very great, which is useless to convergence). Therefore, the mutation times for each new chromosome with encoded numbers of 1-5 is a random integer of 1-8 and with encoded numbers of 6-21 is a random integer of 1-3, which therefore can prevent the generation of vibration as well as can get new solutions different from the current population.

Both the crossover probability and mutation probability of new algorithm is a random number which is exempted from probability determination, thus enhancing the algorithm's practicability and stability.

We know that HS [6] has three parameters: HMS、HMCR、PAR and it perform quite differently with different combinations of parameters. It can search the global optimal in 3 second with the best combination but fail with other combination even. Otherwise, as different puzzle has different best combination, we cannot know it in advance. Similar to HS, GAuGE [5] has to define three parameters: Crossover probability 、 Position fields mutation probability and Value fields mutation probability in advance. GA/ACO [4] has to find the best parameters with a lot of statistic. But our new algorithm uses a random strategy to avoid such parameters and get a stable performance. Obviously, the new algorithm in this paper has a better maneuverability and stability.

Table 2 Mutation operator of new algorithm

```
...
  if fitness(1) > 10
       swap = 1;
   elseif fitness(1) > 6
       swap = randint(1, 1, [1, 5]);
   else
       if identifier < 6
           swap = randint(1, 1, [1, 8])
       else
           swap = randin(1, 1, [1, 3]);
       end
   end
   ...
```

## 4. Simulation experiment and results analysis

### 4.1 The comparison results of new algorithm and IGA

In order to compare the performance of new algorithm and IGA, we randomly drew 14 Sudoku puzzles from website [12] to do an experiment, 10 from the puzzle bank with easy and challenging and 4 from the puzzle bank with difficulty and super difficulty. Use the new algorithm and IGA in [10] to operate 10 experiments respectively for each puzzle with easy and challenging, 200 experiments in all; for 4 puzzles with difficulty and super difficulty, operate 6 experiments respectively for each, 48 experiments in all.

*Table* 3 indicates the results of new algorithm and IGA of the 100 experiments for the 10 puzzles with easy and challenging. Generation of success represented the average generation which the algorithm successfully reach to fitness value "w" (w = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1) in 100 times experiments, representing the convergence of the algorithm for the Sudoku puzzles with easy and challenging. Convergence probability represented the success rate which the algorithm Converge to fitness value "w" (w = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1) in 100 times experiments. When the fitness value is 0, it shows that we get a global optimal solution. On average, the success rate of new algorithm one-time operation is 80%, and IGA is only 17%. It can be seen that convergence probability of new algorithm for the global optimal solution is over 4 times more compared to IGA. From *Table* 3 it is easy to see that new algorithm has a quicker convergence speed than IGA for Sudoku puzzles with easy and challenging. Furthermore, the less the fitness value is, the more obvious tendency is.

*Table* 4 indicates the results of new algorithm and IGA of 24 experiments for 4 puzzles with difficulty and super difficulty. Generation of success represented the average generation which the algorithm successfully reach to fitness value "w" (w = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1) in 24 times experiments, representing the convergence of the algorithm for the Sudoku puzzles with difficulty and super difficulty. Convergence probability represented the success rate which the algorithm Converge to fitness value "w" (w = 10, 9, 8, 7, 6, 5, 4, 3, 2, 1) in 24 times experiments. It can be seen from *Table* 3-4 that new algorithm can solve a global optimal solution for Sudoku faster than IGA with easy and challenging. But with difficulty and super difficulty IGA got stuck to local optima more easily than new algorithm, and it was not able escape from them. So the ability to avoid falling local convergence for new algorithm is better than IGA. The data in *Table* 3 and 4 shows that new algorithm has a quicker convergence speed than IGA and a higher success rates.

Table 3 Convergence process of new algorithm (NA) and IGA in solving easy and challenging rating puzzle

| Statistics \ Fitness | | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Generation of success | NA | 16 | *15* | 21 | 23 | 29 | 49 | 63 | 113 | 162 | 465 |
| | IGA | 26 | 31 | 43 | 45 | 61 | 68 | 87 | 85 | 118 | 189 |
| Convergence probability(%) | NA | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 98 | 80 |
| | IGA | 100 | 100 | 99 | 94 | 90 | 83 | 73 | 47 | 46 | 17 |
| Generation of Convergence | NA | 16 | 15 | 21 | 23 | 29 | 49 | 63 | 113 | 165 | 579 |
| | IGA | 26 | 31 | 43 | 48 | 68 | 82 | 119 | 181 | 256 | 1,112 |

Table 4 Convergence process of new algorithm (NA) and IGA in solving difficulty and super difficulty rating puzzle

| Statistics \ Fitness | | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Generation of success | NA | 7 | 9 | 13 | 14 | 45 | 49 | 52 | 146 | 185 | 1,556 |
| | IGA | 55 | 74 | 72 | 83 | 137 | 160 | 267 | 125 | 808 | —— |
| Convergence probabilit(%) | NA | 100 | 100 | 100 | 100 | 100 | 100 | 96 | 75 | 58 | 8 |
| | IGA | 100 | 100 | 97.5 | 92.5 | 75 | 72.5 | 45 | 25 | 2.5 | —— |
| Generation of Convergence | NA | 7 | 9 | 13 | 14 | 45 | 49 | 54 | 195 | 319 | 19,450 |
| | IGA | 55 | 74 | 90 | 183 | 221 | 593 | 500 | 32,320 | —— | —— |

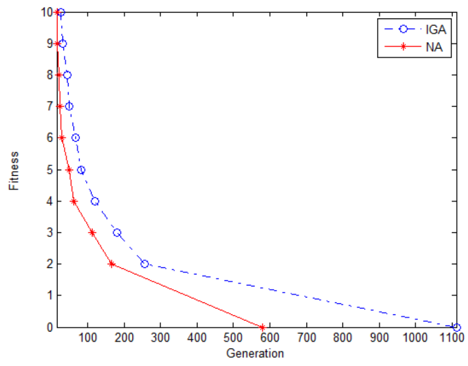


Fig.3 Curve of average convergence in solving easy and challenging rating puzzle

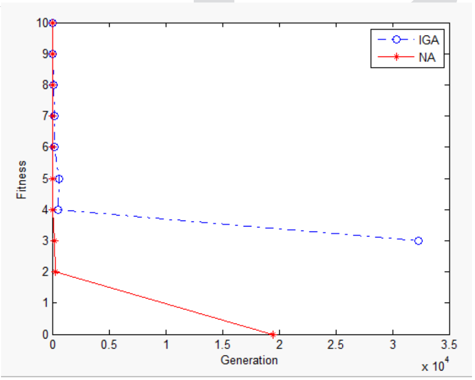


Fig.4 Curve of average convergence in solving difficulty and super difficulty rating puzzle

*Figure* 3 and *Figure* 4 shows the average convergence process of NA and IGA for solving various rating puzzle. From *Figure* 3 it is easy to see that NA has a quicker convergence speed than IGA for Sudoku puzzles with easy and challenging. Furthermore, the less the fitness value is, the more obvious tendency is. From *Figure* 4, we can see that IGA can't research and find the optimal solutions in solving difficulty and super difficulty rating puzzle but NA can.

## 4.2 The comparison results of new algorithm and GA、CA、 ACO、 GA/ACO

We compared the performance of new algorithm with other meta-heuristics algorithm, i.e., GA、CA、ACO and GA/ACO hybrid, by solving 46 different benchmark Sudokus [13].We run each the benchmark Sudoku puzzles 10 times and calculate the average generations and success rate for each one, then comes *Table* 5 and *Table* 6. At the column of "new algorithm" of *Table* 5, "------" means that the algorithm has run more than 10000 generations before solving the Sudoku puzzle. We think that doing more experiments is meaningless if the algorithm need to run more than 10000 generations to solve a Sudoku puzzles. The data of the other column (GA, CA, ACO, GA/ACO) comes from the product of table 8 and table 9 of paper [4], for example, the first data 163 is the product of the first data 192 at table 8 and the first data 0.85 at table 9. Because table 8 is the comparison of how many generations different methods run between reinitilizations with benchmark Sudokus and table 9 is the comparison of the average number of reinitializations needed in order to solve Sudokus with different methods, *Table* 5, as the product of table 8 and table 9, is the comparison of how many generations is needed to solve Sudoku with different methods. The "avg." is the average of all the data of each column except those at the locations corresponding to "------". It can be seen from *Table* 5 that new algorithm outperform GA, CA, and GA/ACO. ACO needed a less generations than the other methods, mainly because it uses a large population size than the other methods.

*Table* 6 is comparison of number of runs, out of 100 runs, that Sudoku puzzles were solved without reinitialization. We get the data of GA, AC, ACO, GA/ACO from table 10 of paper [4], and the data of new algorithm after running each Sudoku puzzles from [13] 10 times. Similar to *Table* 5, we can also get the "avg." at *Table* 6. Because we run each Sudoku puzzles from [13] only 10 times, the number of success is 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10 corresponding to the number <10, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 at *Table* 6. "<10" means new algorithm cannot solve the Sudoku puzzles within 10 trials, in another word, the number of runs, out of 100 runs, that Sudoku puzzles were solved without reinitialization is less than 10. We also think doing more experiments is meaningless if the number is less than 10.

Table 5 Comparison of how many generations is needed to solve Sudokus with different methods

| | GA | | | CA | | | ACO | | | GA/ACO Hybrid | | | New algorithm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | a | b | c | a | b | c | a | b | c | a | b | c |
| 1 | 163 | 81 | 1854 | 179 | 104 | 2290 | 15 | 22 | 88 | 312 | 587 | 1513 | 247 | 611 | 447 |
| 2 | 2474 | 9581 | 2532 | 2856 | 9006 | 3032 | 665 | 324 | 101 | 3465 | 4457 | 1993 | 1988 | 2764 | 1820 |
| 3 | 2861 | 26499 | 7773 | 2873 | 11206 | 7497 | 189 | 1395 | 1124 | 2811 | 8007 | 5450 | 4375 | 9966 | 2857 |
| 4 | 7639 | 18169 | 16167 | 6016 | 18824 | 12796 | 2283 | 1416 | 1295 | 9480 | 11081 | 7909 | 9701 | 4791 | - ----- |
| 5 | 12086 | 9809 | 20131 | 10937 | 8343 | 16129 | 1241 | 4853 | 4473 | 13880 | 6843 | 16616 | ------ | ------- | ------ |
| E | 42 | 37 | 215 | 125 | 55 | 125 | 57 | 124 | 52 | 288 | 427 | 141 | 196 | 62 | 237 |
| C | 21865 | 16045 | 11336 | 5212 | 12388 | 7996 | 449 | 1565 | 423 | 7344 | 14503 | 5316 | 9811 | 9264 | ------ |
| D | 58145 | 22754 | 20773 | 44558 | 20625 | 23653 | 14064 | 2992 | 809 | 32374 | 9780 | 11475 | ------ | ------ | ------ |
| SD | 93832 | 47509 | 54021 | 55054 | 40699 | 37770 | 21678 | 7942 | 10937 | 43530 | 45564 | 39006 | ------ | ------ | 9699 |
| Easy | 4807 | 549 | 418 | 2337 | 477 | 283 | 89 | 26 | 38 | 1720 | 422 | 595 | 2051 | 2174 | 1005 |
| Med. | 13929 | 36807 | 12788 | 12260 | 29831 | 19535 | 750 | 9520 | 2738 | 12340 | 30398 | 6982 | 4762 | ------ | 2058 |
| Hard | 258106 | 13173 | 109721 | 160776 | 14417 | 71208 | 333560 | 5728 | 43779 | 159653 | 17405 | 72102 | ------ | 9727 | ------ |
| GA-E | 715 | 742 | 388 | 453 | 706 | 418 | 36 | 63 | 41 | 774 | 1104 | 671 | 1049 | 1105 | 927 |
| GA-M | 6901 | 3071 | 5260 | 5807 | 2372 | 4725 | 496 | 539 | 1608 | 7068 | 4065 | 6072 | 9947 | 4232 | 2648 |
| GA-H | 36606 | 16282 | 174520 | 31238 | 12236 | 117386 | 8964 | 5374 | 58801 | 27461 | 18585 | 122109 | 9643 | ------ | ------ |
| AI_E | | 460335 | | | 330371 | | | 98228 | | | 244320 | | | ------ | |
| Avg. | | 9039 | | | 7152 | | | 1396 | | | 6997 | | | 4006 | |

Table6 Comparison of number of runs, out of 100 runs, that Sudokus were solved without reinitialization

| | GA | | | CA | | | ACO | | | GA/ACO Hybrid | | | new algorithm | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | a | b | c | a | b | c | a | b | c | a | b | c | a | b | c |
| 1 | 58 | 68 | 48 | 56 | 76 | 46 | 70 | 54 | 41 | 48 | 53 | 42 | 100 | 100 | 100 |
| 2 | 41 | 22 | 31 | 42 | 22 | 22 | 10 | 23 | 30 | 21 | 26 | 27 | 40 | 30 | 40 |
| 3 | 34 | 15 | 16 | 42 | 19 | 17 | 22 | 6 | 5 | 18 | 11 | 5 | 20 | 10 | 30 |
| 4 | 9 | 5 | 11 | 7 | 7 | 13 | 4 | 7 | 3 | 7 | 5 | 7 | 10 | 20 | <10 |
| 5 | 5 | 12 | 10 | 4 | 8 | 9 | 4 | 1 | 0 | 6 | 10 | 0 | <1 0 | <10 | <10 |
| E | 82 | 83 | 77 | 72 | 86 | 69 | 74 | 54 | 79 | 44 | 67 | 59 | 100 | 100 | 100 |
| C | 33 | 10 | 17 | 27 | 8 | 22 | 12 | 2 | 21 | 31 | 3 | 23 | 10 | 10 | <10 |
| D | 6 | 11 | 22 | 11 | 13 | 25 | 0 | 5 | 14 | 6 | 18 | 15 | <1 0 | <10 | <10 |
| SD | 5 | 8 | 5 | 4 | 8 | 3 | 0 | 2 | 1 | 2 | 3 | 5 | <10 | <10 | <10 |
| Easy | 34 | 62 | 50 | 39 | 60 | 55 | 58 | 62 | 48 | 37 | 53 | 39 | 40 | 40 | 70 |
| Med. | 10 | 2 | 29 | 13 | 2 | 26 | 11 | 1 | 3 | 15 | 4 | 25 | 20 | <10 | 40 |
| Hard | 2 | 10 | 4 | 3 | 6 | 1 | 0 | 0 | 2 | 0 | 3 | 4 | <10 | 10 | <10 |
| GA-E | 37 | 52 | 35 | 28 | 52 | 36 | 43 | 39 | 16 | 23 | 29 | 16 | 60 | 70 | 70 |
| GA-M | 14 | 11 | 15 | 11 | 9 | 8 | 7 | 9 | 3 | 11 | 9 | 9 | 10 | 20 | 30 |
| GA-H | 3 | 5 | 4 | 0 | 4 | 1 | 0 | 0 | 0 | 2 | 7 | 2 | 10 | <10 | <10 |
| AI_E | | 5 | | | 3 | | | 0 | | | 1 | | | <10 | |
| Avg. | | 34.27 | | | 33.13 | | | 27.3 | | | 41.62 | | | 45.17 | |

From *Table* 5 and *Table* 6, although the average generations needed to solve Sudoku puzzles is more than ACO, the number of runs, out of 100 runs, that Sudoku puzzles were solved without reinitialization is the best. So the new algorithm performs quite well comparing with GA, AC, ACO, GA/ACO (not include difficult level Sudoku puzzles).

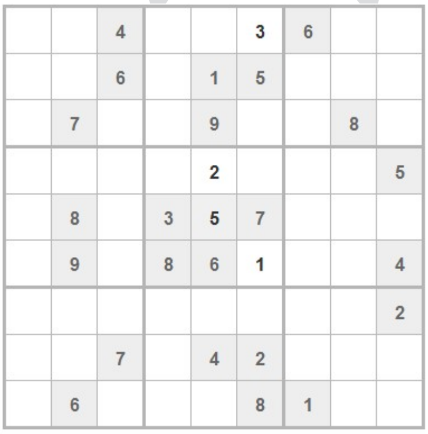
## 4.3 The comparison results of new algorithm and HS、GauGE

In order to compare the performance of new algorithm and HS, new algorithm was applied to solving the Sudoku puzzles taken from *Fig*. 1(easy level) and *Fig*. 3(hard level) of Geem [6]. *Table* 7 presents solve times of the easy Sudoku puzzle. It can be seen that the average solve time of new algorithm is only 1.75 seconds without defining any parameters in advance. But from *Table* 1 of Geem [6], we can see that HS performs quite sensitive to the parameters of HMS、HMCR、PAR. The best combinations of parameters are different for different Sudoku puzzle, even the best combination would cost 3 seconds to solving the Sudoku puzzles taken from *Fig*. 1(easy level) of Geem [6]. In the case of the hard Sudoku puzzle with 26 given values, HS is instead entrapped in one of local optimal with penalty of 14 after 1064 function evaluations. But new algorithm can search the global optimal in 20% or got one of local optima with penalty of 10 after 811 function evaluations. Obviously new algorithm has a better result than HS.

To solving the Sudoku puzzles with one solution only ( well formed puzzles), GAuGE use five strategies: Last remaining、Slice and Dice、Column Fill、Row Fill、Raising Numbers. But only use this five strategies cannot always fill all the spaces of all the well formed puzzles purely on logic. For example, *Figure* 5 shows a well formed puzzle and *Figure* 6 show the process of solving Sudoku of *Fig*.5 with the five strategies above. The big numbers is the given numbers and the little numbers is the remaining numbers which fulfill the five strategies above. We can see that each space has two remaining numbers at least and we cannot fill space any more. From *Table* 2 of Nicolau and Ryan [5] it is easy to see that the puzzles #116 and #117 were never solved. So GAuGE can only solve a part of Sudoku puzzles. However, new algorithm can solve all kind of Sudoku puzzles with quite a fast speed.

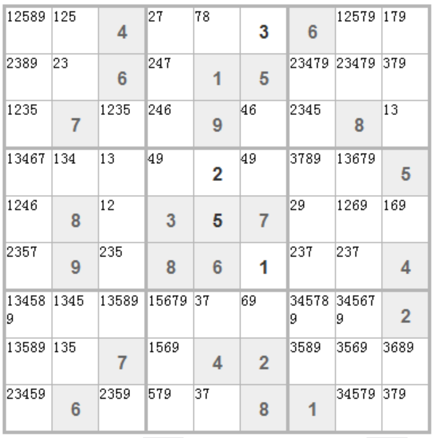
Table 7 10 runs of solving easy Sudoku with new algorithm

| Number | Iterations | Times |
|---|---|---|
| 1 | 40 | 1.57 |
| 2 | 60 | 2.18 |
| 3 | 51 | 1.79 |
| 4 | 73 | 2.46 |
| 5 | 32 | 1.20 |
| 6 | 69 | 2.45 |
| 7 | 28 | 1.07 |
| 8 | 48 | 1.74 |
| 9 | 29 | 1.10 |
| 10 | 41 | 1.45 |
| Average | **48.3** | **1.75** |



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 4 | | | 3 | 6 | | |
| | | 6 | | 1 | 5 | | | |
| | 7 | | | 9 | | | 8 | |
| | | | | 2 | | | | 5 |
| | 8 | | 3 | 5 | 7 | | | |
| | 9 | | 8 | 6 | 1 | | | 4 |
| | | | | | | | | 2 |
| | | 7 | | 4 | 2 | | | |
| | 6 | | | | 8 | 1 | | |

Fig.5 A hard level Sudoku with 27 given numbers



| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 12589 | 125 | 4 | 27 | 78 | 3 | 6 | 12579 | 179 |
| 2389 | 23 | 6 | 247 | 1 | 5 | 23479 | 23479 | 379 |
| 1235 | 7 | 1235 | 246 | 9 | 46 | 2345 | 8 | 13 |
| 13467 | 134 | 13 | 49 | 2 | 49 | 3789 | 13679 | 5 |
| 1246 | 8 | 12 | 3 | 5 | 7 | 29 | 1269 | 169 |
| 2357 | 9 | 235 | 8 | 6 | 1 | 237 | 237 | 4 |
| 134589 | 1345 | 13589 | 15679 | 37 | 69 | 345789 | 345679 | 2 |
| 13589 | 135 | 7 | 1569 | 4 | 2 | 3589 | 3569 | 3689 |
| 23459 | 6 | 2359 | 579 | 37 | 8 | 1 | 34579 | 379 |

Fig.6 The remain numbers in each space

## 5. Conclusions

In this paper, a new effective genetic algorithm is proposed. The selection operator, crossover operator and mutation operator of the genetic algorithm have effectively been improved according to features of Sudoku puzzles. We described in detail random search strategy of new algorithm and studied performance of new algorithm. Analysis of the results revealed that new algorithm outperformed others meta-heuristics algorithm in solving Sudoku puzzles. This might be due to the fact that the cross-point position, crossover probability and mutation probability of new algorithm is a random number and new algorithm uses group as a division for priority level in selection operator and under the circumstance of no better chromosome achieved after 4 times of iteration it should be replaced by a near-optimal chromosome in time, so as to make the searching direction for each iteration searching process is more flexible and the searching scope is wider and population gene is more diversity and the probability of chromosome having more abundant genes for selection is reasonably enlarged and the evolution direction of population can be change timely. A large number of simulation experiment results showed that new algorithm is superior to the other meta-heuristics algorithm in terms of success rates and convergence speed.

## 6. Acknowledgement

This research was financially supported by the Science Technology Project of Guangdong Proving (Grant NO.2011B010200042) ，the Natural Science Foundation of Guangdong Proving (Grant NO.S2011040004804, No.10251009001000002) and the National Natural Science Foundation of China (Grant NO.61100148，No.60974077).

## References

[1] I. Semeniuk, Stuck on you, New Scientist, vol. 31, pp. 45-47, 2005.

[2] http://en.wikipedia.org/wiki/Sudoku. Accessed 16 Oct 2006.

[3] F. Sullivan, Born to compute. Comput. Sc. Eng. vol. 8, pp. 88-90, 2006.

[4] T. Mantere, J. Koljonen, Ant Colony Optimization and a Hybrid Genetic Algorithms for Sudoku Solving, In: 15th International Conference on Soft Computing, Brno, Czech Republic, Mendell, pp.41-48, 2009.

[5] M. Nicolau, C. Ryan, Solving Sudoku with the GAuGE System. In: Collet, P., Tomassini, M., Ebner, M., Gustafson, S., Ekárt, A. (eds.) EuroGP 2006. LNCS, vol. 3905, pp. 213-224,2006.

[6] Z.W. Geem, Harmony search algorithm for solving sudoku, Lect. Notes Comput, LNAI , vol.4692, pp.371-378, 2007.

[7] M. Perez, T. Marwala, Stochastic Optimization Approaches for Solving Sudoku. In: Computer Science - Neural Evolutionary Computing, arXiv: 0805.0697v1. http://arxiv.org/ftp/arxiv/papers/ 0805/0805.0697.pdf, 2008.

[8] T. Mantere, J. Koljonen, Solving, Rating and Generating Sudoku Puzzles with GA. 2007 IEEE Congress on Evolutionary Computation-CEC2007,Singapore, pp.1382-1389, 2007.

[9] T. Mantere, J. Koljonen, Solving and analyzing Sudokus with cultural algorithms. 2008 IEEE Congress Computational Intelligence - WCCI2008, 1-6 June, Hong Kong, China, pp. 4054-4061, 2008.

[10] Y.D. Li, X.Q. Deng, Solving Sudoku puzzles base on improved genetic algorithm. Comput. Appl. Softw. , vol. 28(3), pp.68-70, 2011.

[11] X.D. Duan, C.R. Wang, X.D. Liu, Particle Swarm Optimization and Application, Liao Ning University Press, Shen Yang , pp.145-152, 2007.

[12] http://www.llang.net/sudoku/

[13] T. Mantere, J. Koljonen, Sudoku project homepage,
Available via. http://lipas.uwasa.fi/~timan/sudoku/