Optim Lett (2013) 7:241–257
DOI 10.1007/s11590-011-0413-0

ORIGINAL PAPER

A novel hybrid genetic algorithm for solving Sudoku puzzles

Xiu Qin Deng · Yong Da Li

Received: 9 September 2010 / Accepted: 11 October 2011 / Published online: 26 October 2011

Abstract In this article, a novel hybrid 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. The improved selection operator has impaired the similarity of the selected chromosome and optimal chromosome in the current population such that the chromosome with more abundant genes is more likely to participate in crossover; such a designed crossover operator has possessed dual effects of self-experience and population experience based on the concept of tactfully combining PSO, thereby making the whole iterative process highly directional; crossover probability is a random number and mutation probability changes along with the fitness value of the optimal solution in the current population such that more possibilities of crossover and mutation could then be considered during the algorithm iteration. The simulation results show that the convergence rate and stability of the novel algorithm has significantly been improved.

Keywords Genetic algorithm · Sudoku puzzles · Convergence rate · Optimization

1 Introduction

Genetic algorithm (GA) [3] is an optimization method for self-organization and self-adaption concerning simulation of evolution process of species in Nature and

X. Q. Deng (✉)
Faculty of Applied Mathematics, Guangdong University of Technology,
Guangzhou 510006, Guangdong, People's Republic of China
e-mail: xiuqindeng@163.com

Y. D. Li
Tencent Technology Company Limited, Shenzhen 518057, Guangdong, People's Republic of China

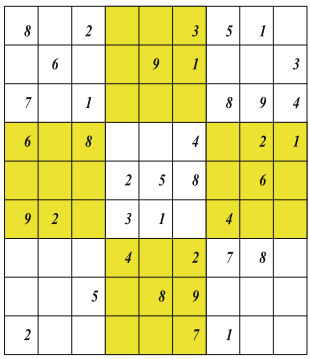

Fig. 1 A starting point of the Sudoku puzzle, where 38 locations contains a static digit that are given

8		***2***			***3***	***5***	***1***	
	6			***9***	***1***			***3***
7		***1***				***8***	***9***	***4***
6		***8***			***4***		***2***	***1***
			2	***5***	***8***		***6***	
9	***2***		***3***	***1***		***4***		
			4		***2***	***7***	***8***	
		5		***8***	***9***			
2					***7***	***1***		

Fig. 2 A solution for the Sudoku puzzle given in Fig. 1, The given numbers marked in *bold*

8	9	***2***	7	4	***3***	***5***	***1***	6
5	***6***	4	8	***9***	***1***	2	7	***3***
7	3	***1***	6	2	5	***8***	***9***	***4***
6	5	***8***	9	7	***4***	3	***2***	***1***
1	4	3	***2***	***5***	***8***	9	***6***	7
9	***2***	7	***3***	***1***	6	***4***	5	8
3	1	9	***4***	6	***2***	***7***	***8***	5
4	7	***5***	1	***8***	***9***	6	3	2
2	8	6	5	3	***7***	***1***	4	9

mechanism of decomposing problems, and particle swarm optimization (PSO) algorithm [6] is an optimal technology based on swarm intelligence. Both GA and PSO have widely been applied in computer science, artificial intelligence, IT and engineering practice. This article studies if the Sudoku puzzles can be solved effectively with a novel genetic algorithm.

Sudoku puzzles are composed of a 9×9 grid, namely of 81 positions, which are then divided into nine 3×3 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 3×3 sub-grids contains each integer $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ once and only once [4]. Figure 1 shows a Sudoku puzzle example having a unique solution [11] for testing this article's algorithm. It contains 38 givens, the corresponding solution for Sudoku puzzle can be achieved by filling up the remaining 43 empty spaces. The solution of this Sudoku is shown in Fig. 2. Note that givens has remained in their original positions.

Sudoku puzzles has been claimed to be very popular and even addictive because they are very challenging but have relatively simple rules [17]. Playing Sudoku puzzles on 12 November 2004 was first posted inside of a newspaper 'Times", becoming a daily fixed content of this newspaper, and later it become hugely popular all over the

world, particularly in America and Europe [18]. At present, the research on Sudoku puzzles focuses mainly on two aspects: minimum number of given numbers in Sudoku puzzles having a unique solution and method of solving Sudoku puzzles having a unique solution. What are the possible minimal grids required upon which Sudoku puzzles would lose any solution even if you move any one of numbers out of the grids. This problem is one of the most recreational mathematics in Sudoku puzzles; unfortunately, the problem mentioned herein has not been solved so far. However, as estimated by mathematicians this figure could be 17 [4]. For those resolutions of Sudoku puzzles having a unique solution the strategy applied for manually solving the puzzles would cover Singles Candidature, Hidden Singles Candidature and Naked Pairs, etc. As for manually solving the puzzles the solution integrated with the help of computer would be valuable in the research. Many methods regarding the solving Sudoku puzzles with the help of computer have been put forward.

Nicolau and Ryan [14] have used quite a different approach to solve Sudoku: their Genetic Algorithms using Grammatical Evolution (GAuGE) optimizes the sequence of logical operations that are then applied to find the solution. They [15] 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. To solve 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 can't fill all the spaces of all the well formed puzzles purely on logic. Solving well formed puzzles, you will often find each space has two fit numbers at least, then you can't solve the solution using the strategies above only. So GAuGE can't find the solution of #116 and #117 in [15] forever.

Moraglio et al. [9] have solved Sudoku puzzles using genetic algorithm with product geometrical crossover. Their method solves easy Sudoku efficiently but it has limited efficiency for those Sudoku puzzles with medium and superior difficulties. Moraglio et al. [10] also used the geometric particle swarm optimization (GPSO) for solving Sudoku puzzles, but their results are not so good as those by genetic algorithm. The success rate of one-time operation by the former is only 14–72% while it is 100% by the latter.

Geem [2] 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. Moreover HS perform quite sensitive to the parameters of HMS, HMCR, PAR.

Li [7] introduced graph search strategy on the basis of knowledge representation and deduction base in artificial intelligence for solving Sudoku puzzles.

Perez and Marwala [16] 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 [11–13] 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.

Mantere and Koljonen [11], however, proposed a new idea for solving Sudoku puzzles based on genetic algorithm which is innovatively valuable and novelty as well in the light of research, but with a slow rate of convergence. In order to enhance the convergence speed of the genetic algorithm, Li and Deng [8] 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) [8] 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 article effectively improves the selection operator, crossover operator and mutation operator. Furthermore, the information exchange pattern of particle swarm optimization is tactfully used in the crossover operator, such a designed crossover operator has possessed dual effects of self-experience and population experience, making the whole iteration process more directional.

2 Genetic algorithm for solving Sudoku puzzles

2.1 Fitness function

The key point for solving Sudoku puzzles by successfully using genetic algorithm is how to construct a fitness function and how to encode a solution (chromosome)of Sudoku puzzles. In addition to basic rules of Sudoku, the givens must be observed during the solving process. Therefore the solution of a Sudoku puzzle must be satisfied with the following four conditions: (1) each row has to contain each integer from 1 to 9; (2) each column has to contain each integer from 1 to 9; (3) each 3×3 sub-grids must contain each integer from 1 to 9; (4) the given numbers must stay in the original positions. By selecting an appropriate solving strategy, condition (4) is always fulfilled. Additionally, one of the condition (1) to (3) can be controlled. Hence, only two conditions are subject to optimization. We chose to implement our GA so that condition (3) and (4) are intrinsically fulfilled and only the condition (1) and (2) are optimized. Thus, for Sudoku puzzle shown in Fig. 1 the remained 43 empty spaces are randomly filled up under the precondition of satisfying the condition (3) and (4) so as to achieve a feasible solution of Sudoku puzzle. Such a feasible solution is not necessary to satisfy the conditions (1) and (2), the fitness function is defined according to the content of satisfying both conditions as follows:

$$w = \sum_{i=1}^{9} (r_i + c_i) \tag{1}$$

Fig. 3 Encoding chromosome corresponding to a given Sudoku puzzle

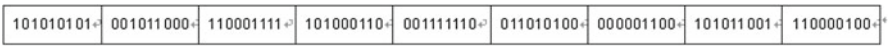

Fig. 4 Associated chromosome corresponding to a given Sudoku puzzle

where r_i is number of repeated integer on the i row while c_i is numbers of repeated integer on the i column. The value of w is non-negative integer, the smaller the value of w is, the higher the fitness of the corresponding feasible solution, thus when w is 0, the corresponding feasible solution becomes optimal solution for the given Sudoku puzzle.

2.2 Encoding chromosome

Encoded numbers each grid of Sudoku puzzle in an order from left to right and from top to bottom, the Sudoku puzzle are then encoded according to a sequence from a smaller encoded number to a larger encoded number [11], so as to express the feasible solution (chromosome) of Sudoku puzzle using an integer array of 81 numbers, which is then divided into nine sub-blocks of nine numbers, each sub-block corresponding to one 3×3 sub-grid from left to right and from top to bottom, accordingly. The corresponding chromosomes in Fig. 1 are shown in Fig. 3.

In which zero represents a number ready for filling up, non-zero is the given numbers in the sub-grid. Number corresponding to non-zero location would be changed into 1.The correspondent associated chromosome could be achieved as shown in Fig. 4. The associated chromosome is used for checking fixed positions, if there is a number that is not equal to zero that number cannot be changed during the optimization, only the positions that have zero are free to be changed, thereby ensuring unchangeable of the original Sudoku puzzle in the procedure of mutation.

2.3 Genetic operator analysis

Both cross-point position and mutation probability of IGA [8] are constant that means any change of Sudoku puzzles both crossover probability and mutation probability must be redefined, thus to affect the stability and practicability of the algorithm. Small mutation probability in the later period of iterator in [8] cannot keep the abundant of gen. Thus, once the solutions fall in the nearby of a local optimal solution, the algorithm will fall into local optimal solution easily (see in Figs. 7, 8). Finally the selection operator as given by Li and Deng [8] based on the principle of the survival of the fittest would make the probability of selecting chromosome with smaller encoded number as parental type too high. All the operator of the IGA above will make the algorithm easily fall in local optimal solution.

On the basis of analysis above each genetic operator of IGA will further been improved according to characteristic of Sudoku puzzles in this article. In order to avoid

the problem of determining the probability of crossover and mutation, a "random" technology is used in this article (see Sects. 3.2, 3.3 for more detail). Also two strategies: "divide into group" and "add final fitness value to the optimal solution of present generation" are used in the selection operator to reduce the probability of falling into local optimal solution and change the direction of evolution in time (see Sect. 3.1 for more detail). Furthermore, the information exchange pattern of particle swarm optimization is used in the crossover operator, such a crossover operator has possessed dual effects of self-experience and population experience, making the whole iteration process more directional. As the genetic algorithm designed in this paper integrates with the principle of the particle swarm optimization, the new algorithm is called hybrid genetic algorithm (HGA).

3 Hybrid genetic algorithm

A feasible solution for a given Sudoku puzzle could be achieved by randomly filling it up under the precondition of satisfying condition (3). Repeat this procession to achieve 41 feasible solutions, which corresponds to 41 chromosomes, choose the best 21 chromosomes, thus the initialization of population is completed.

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 $w_i, i \in \{1, 2, 3 \ldots 21\}$. 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.

Where the value of "identifier" is a newly generated chromosome, presently. "Upper_bound" is a variable, "randint(1,1,[2,Upper_bound])" will generate a random integer in a range of 2-Upper_bound, "$acc(1) = acc(1) + 0.25$" represents the increase of fitness by 0.25 for optimal chromosome in the current population before selection conducted (namely a chromosome with the encoded number 1), remaining the chromosome into the next generation. 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. In order to avoid falling of local optimal with algorithm 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 representative of genes of the population is changed in crossover operation, namely evolutionary direction of population genes is thereby changed, too.

Table 1 Selection operator of HGA

```
...
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
...
```

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 $k = 2, 3, \ldots, 21$ is an encoded number of chromosome). Any chromosome in each group is selected as a parental generation with times being 6/20 +5/15+5/10+5/5, 6/20 +5/15+5/10, 6/20 +5/15, 6/20, respectively. 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. In addition, two chromosomes selected as a parental type would finally become a single chromosome. When X_1 and X_2 as parental types are identical, the selection operator would be repeated until both chromosomes become different.

3.2 Crossover operator

The action of crossover operator is to search the better combination of genes, i.e. chromosome, on the basis of current population genes. Crossover and mutation operators cannot directly be used for solving Sudoku puzzles using genetic algorithm since an illegal state would be generated from 3 × 3 sub-blocks. Multiple point crossover is applied in this article. The cross-points corresponding a 3 × 3 sub-block in the same location would exchange randomly between two selected chromosome (solution). 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

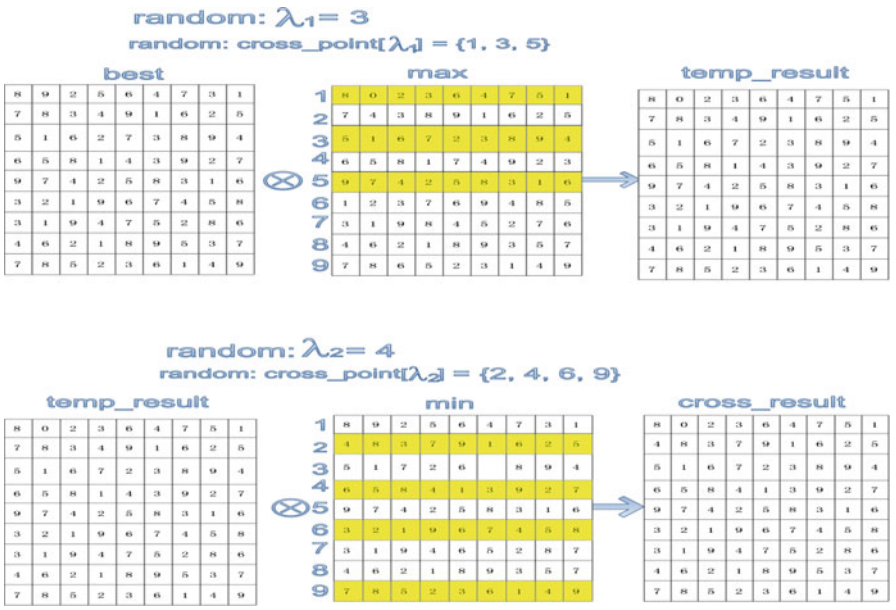

Fig. 5 The process of crossover

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.

Although a better chromosome can be achieved from crossover among the better chromosomes in the view of Probability Statistics, this tendency is, however, not obvious. 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 [1], in this article, this information exchange mode of PSO is applied in crossover operator of GA, the renewal of chromosome is affected by dual effects of self-experience and population experience, thereby making the crossover operator highly directional. In this article the self-experience is originated from two selected parental chromosomes, the optimal chromosome in the current population represents the population experience. 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), λ_1 and λ_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 λ_1, thus the crossover operator could be represented as follows:

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

Figure 5 indicates the implementing process of crossover operator. For the random natural numbers λ_1, λ_2 which are in the range of 1–8 and the three selected parental chromosomes (best, max and min), first produce λ_1 cross-points (i.e. λ_1 sub-blocks)

randomly in max. For the sub-blocks in these cross-points, get offspring temp-result directly from the parental chromosome (max); for the sub-blocks of offspring temp-result at other positions, select them from the sub-blocks at the positions corresponding to the parental chromosome (best) in proper order. For offspring temp-result and parental chromosomes (min), likewise, first produce λ_2 cross-points randomly in min; for the sub-blocks in these cross-points, get offspring cross- result directly from the parental chromosome (min); for the sub-blocks of the offspring cross-result at other positions, select them from the sub-blocks at the positions corresponding to the offspring temp-result.

In order to make the chromosome formed with crossover operation diversity the crossover operation should be carried out repeatedly if the chromosome is identical to the parental types with crossover operator until a new chromosome appears. 20 new chromosomes will be achieved after 20 pairs of chromosomes crossed over.

3.3 Mutation operator

The action of mutation operator is to make genes of chromosome more abundant during evolutional process, thus to reduce the probability of falling local search. The mutation operation is performed as swap of two points inside a sub-block to make sure each 3×3 sub-blocks is satisfied with condition (3) all along. The associated chromosome as shown in Fig. 4 is used to check if the position is appropriate for mutation. If the digit is not a zero, the corresponding location is illegal to exchange, thus the given digits will not changed during optimization. Because the difference of initialized genes is relatively larger at the early stage of evolution the mutation probability is not necessarily larger. Under the action of crossover operator the integral population fitness gradually becomes smaller, and chromosomes become more and more identical, in this case an adequate increase of probability of mutation will avoid the population to be precocious. However, when the adaption of population decreases again the population would still appear precocious, it will cause vibration of population fitness if the probability of mutation becomes larger again. Therefore, the probability of mutation of chromosome with encoded numbers of 2–6 at this article is enlarged to enlarge the distance between chromosomes and with encoded numbers of 7–21 is diminished to decrease the distance between chromosomes, thereby control the vibration of population fitness within a certain range and avoid population precocious while introducing more abundant genes. The detailed mutation probability is shown in Table 2. Figure 6 indicates the implementing process of mutation operator.

Where swap represents digit of mutation resulted from each chromosome, identifier is an encoded number of chromosome in mutational operation. $randint(1, 1, [1, 5])$, a random integer of 1–5 generated.

4 Step-by-step description of the HGA

The procedure of the HGA can be described in the following:

Table 2 Mutation operator of HGA

```
...
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
...
```

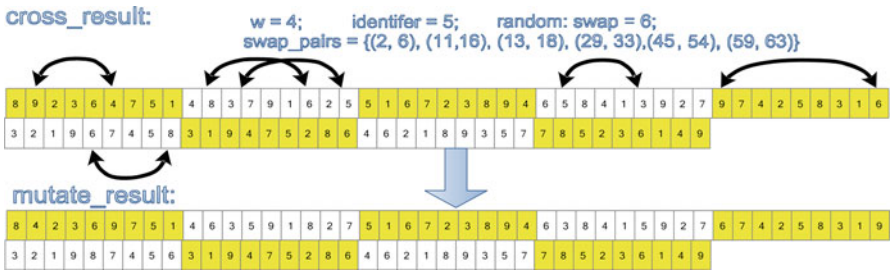

Fig. 6 The process of mutation

Step1 (*Initialization*) The initialization process is repeated to achieve 41 feasible solutions for a given Sudoku puzzle by randomly filling it up under the precondition of satisfying condition (3). Set the current generation $G = 0$.

Step2 (*Encoding chromosome*) Chromosome (feasible solutions) are encoded according to the method introduced in Sect. 2.2.

Step3 (*Calculate fitness and sort*) Calculate fitness of the chromosome according to formula (1) and all of 41 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 41.

Step4 (*Selection*) The 21 chromosomes with encoded numbers in the range of 1–21 are selected as a first generation of population, and the selection operator shown in Table 1 was performed to generate the offspring.

Step5 (*Crossover*) Perform crossover operator shown in formula (2).

Step6 (*Mutation*) Perform mutation operator shown in Table 2.

Step7 $G = G + 1$

Step8 (*Termination criterion*) If $w_i = 0$, stop. Otherwise, go to step 3.

5 Simulation experiment and results

5.1 The comparison results of HGA, IGA and GA

In this section, the performance of HGA is studied by solving Sudoku puzzles shown in Fig. 1 with solution as shown in Fig. 2.The main task in the simulation experiments was to test the convergence performances of each algorithm, and HGA is compared with IGA [8] and GA [11]. The simulation experimental results are shown in Table 3. It can be seen from Table 3 that HGA can solve a global optimal solution for Sudoku faster than IGA and GA in solving easy rating puzzle.

5.2 The comparison results of HGA and HS

In order to compare the performance of HGA and HS, HGA was applied to solving the Sudoku puzzles taken from Fig. 1 (easy level) and Fig. 3 (hard level) of Geem [2]. Table 4 presents solve times of the easy Sudoku puzzle. It can be seen that the average solve time of HGA is only 1.75 s without defining any parameters in advance. But from Table 1 of Geem [2], 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. In this case, even the best combination will cost 3 s. 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 1,064 function evaluations. But HGA can

Table 3 Comparison results of HGA, IGA and GA

Algorithm type	Optimal solution	Generations
HGA (in this article)	See Fig. 2	196
IGA (in literature [8])	See Fig. 2	550
GA (in literature [11])	See Fig. 2	3,500

Table 4 Ten runs of solving easy Sudoku with HGA

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

search the global optimal in 20% or got one of local optima with penalty of 10 after 811 function evaluations. Obviously HGA has a better result than HS.

5.3 Analysis on convergence of the algorithm

We randomly drew 14 Sudoku puzzles from website http://www.llang.net/Sudoku/ [5] to do an experiment, 10 from the puzzle bank with ordinary difficulty and 4 from the puzzle bank with difficulty. Each puzzle has a unique solution and all the puzzles are respectively marked as four different difficulty rating: Easy, Challenging, Difficulty and Super difficult (see Table 5). Use the HGA in Sect. 4 above to operate 10 experiments for each puzzle of the former 10, 100 experiments in all; for the latter 4, operate 6 experiments for each, 24 experiments in all.

Table 5 gives the generations needed by each experiment algorithm convergence. From the experiment data of Table 5, we know that for the 10 Sudoku puzzles with easy and challenging, the success rate of HGA one-time operation is 60–100%, with the average level as 80%, so the success rate is rather high. But the success rate of GPSO [10] one-time operation is only 14–72%.

Table 6 indicates the results of the 100 experiments for the 10 puzzles with easy and challenging, with each line standing for the average convergence process of the results of 10 experiments for each puzzle. The last line shows the average convergence of the average results of the 10 puzzles, representing the convergence of the algorithm for the Sudoku puzzles with easy and challenging. When the fitness value is 0, it shows that we get a global optimal solution, and the average generation for the global optimal solution is 465 generations.

Table 7 indicates the results of 24 experiments for 4 puzzles, with each line standing for the average convergence process of the results of 6 experiments for each puzzle. The last line shows the average convergence of the average results for four puzzles, representing the algorithm's convergence for the Sudoku puzzles with difficulty and super difficulty. The success rate of one-time operation for the four difficult puzzles is lower than 20%, but HGA can always operate to the level 2, with average 648 generations needed. The data in Tables 6 and 7 shows HGA has a quicker convergence speed.

Table 8 indicates the comparison results of HGA, IGA and GA for solving Sudoku puzzles with different difficulty rating. From Table 8, we can see that HGA is better than IGA and GA in statistical for any difficulty rating puzzle. So the operator of this article has popularity for Sudoku puzzles.

Figure 7 shows the average convergence process of HGA and IGA in solving easy and challenging rating puzzle. From Fig. 7 it is easy to see that HGA 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.

Figure 8 shows the average convergence process of HGA and IGA in solving difficulty and super difficulty rating puzzle. From Fig. 8, we can see that HGA and IGA can't research and find the optimal solutions because the puzzles with difficulty and super difficulty have more local convergence point. But HGA can research less fitness

Table 5 The comparison of how effectively HGA finds optimal solution for the Sudoku puzzles with different difficulty rating in 10 times

Difficulty rating	Puzzles	Generations needed when fitness value is zero in each times										Min	Average	Max	Success rate (%)
Easy	1	724	459	141	91	92	59	278	59	63	88	59	205	724	100
	2		61	40	517	488	107	120		944	86	40	295	944	80
	3	97	335	182	565	85	725	721	580	426	356	85	357	725	100
	4	48	405	679	389	484	477	113	369	647	81	48	324	679	100
	5	151			264	204	408	1,623	225	104	808	104	473	1,623	80
	6	938	1,173	223	85	46	1,234	407			572	46	468	1,234	80
Challenge	7		79	96	1,045	63	81	1,960		486	548	63	542	1,960	80
	8	158		305	1,016		725		1,164		443	158	614	1,164	60
	9	742	1,530	1,730				320		242	186	186	792	1,730	60
	10	293		556		169		1,040	412	985		169	575	1,040	60
Difficult	11			1,906								1,906	1,906	1,906	17
	12					1,205						1,205	1,205	1,205	17
Super	13														0
Difficult	14														0

Table 6 Convergence process of HGA in solving easy and challenging rating puzzle

0	10	9	8	7	6	5	4	3	2	1	0
1	15	41	32	22	32	34	55		59		205
2	13	26	19	22	35	56	68	130	94		295
3	11	12	12	28	37	52	61	86	116		357
4	11	9	11	16	20	17	45	135	129		324
5	15	11	21	18	24	36	54	90	355		473
6	8	9	13	10	15	38	23	41	71		468
7	12	10	14	16	19	21	35	84	67		542
8	10	11	16	17	17	22	59	121	138		614
9	20	27	29	34	26	93	134	164	210		792
10	27	32	33	36	51	92	106	267	341		575
Statistical	14	19	20	22	28	46	64	112	158		465

Table 7 Convergence process of HGA in solving difficulty and super difficulty rating puzzle

0	10	9	8	7	6	5	4	3	2	1	0
11	29	60	68	83	336	339	387		896		1,906
12	24	37	33	52	92	159	279		319		1,205
13	35	20	75	63	156	161	201	460	720		
14	33	31	54	53		81	115	352	657		
Statistical	31	37	58	48		185	246	631	648		

Table 8 Comparison results of HGA, IGA and GA

Algorithm type	E, C	D, SD	Population size
HGA (in this article)	579	19,450	21
IGA (in literature [8])	1,112	–	21
GA (in literature [11])	839–9,100	52,211	21

value than IGA. So the ability to avoid falling local convergence for HGA is better than IGA.

From the experiments above done, we can conclude as below: First, this article takes a 3×3 sub-block as a crossover point, with the crossover probability as a random number. In this way, the searching direction for each iteration searching process is more flexible and the searching scope is wider, making up for the shortage that it is difficult to search and find the more optimal solution due to the sparse solution space in this article (the convergence process shown in Tables 6 and 7 indicates that the crossover operator can find the more optimal solution more quickly). Secondly, the crossover probability is a random number which is exempted from probability determination, thus enhancing the algorithm's practicability. Finally, the mutation probability in this article changes along with the fitness value of the optimal solution in the current pop-

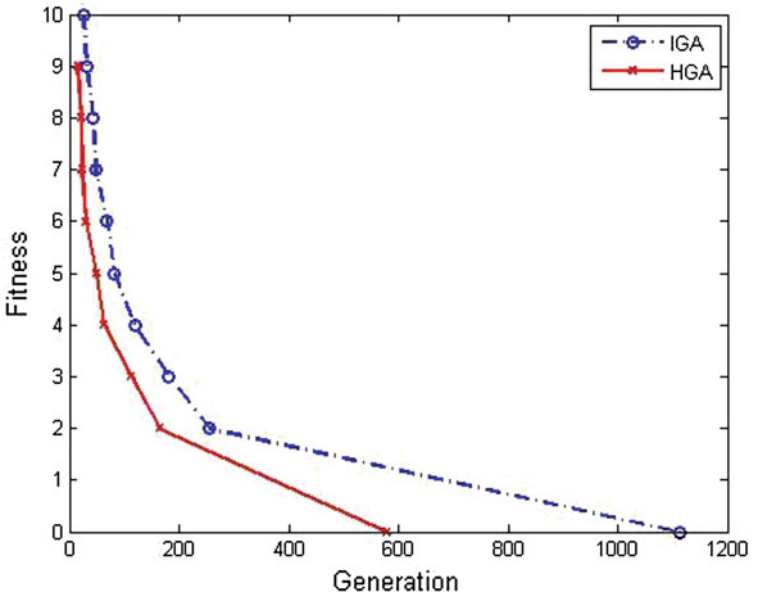

Fig. 7 Curve of average convergence in solving easy and challenging rating puzzle

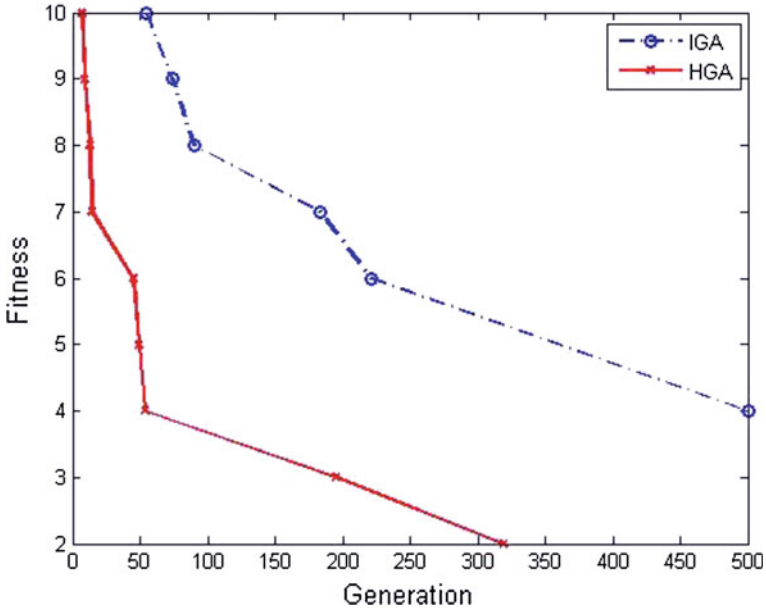

Fig. 8 Curve of average convergence in solving difficulty and super difficulty rating puzzle

ulation in order to maintain the diversity of population gene, which makes it possible for the crossover operator to search and find the solution different with the current population and greatly enhance HGA algorithm's stability for different Sudoku puzzles, and greatly speed up the algorithm's convergence speed.

6 Conclusions and future work

The selection operator, crossover operator and mutation operator of the IGA are further improved with respect to drawback of IGA. First, the novel HGA uses group as a division for priority level in selection operator, thereby impairing the similarity of the selected chromosome and optimal chromosome, so that the probability of chromosome having more abundant genes for selection is reasonably enlarged; secondly, the crossover operator has been endowed with dual effect of self-experience and population experience based on the concept of combining particle swarm optimization, thereby making the whole iteration directional; the secondary probability of mutation will increase along with decrease of fitness, particularly at a later stage of iteration, a reasonable adjustment for mutation probability is conducted according to the fitness value of the optimal chromosomes in the current population, upon which the algorithm reliability could greatly be improved. Finally, 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 change evolution direction of population timely, thereby to avoid falling local problem.

Simulation experimental results showed that not only the novel algorithm can accurately solve a global optimal solution, but also significantly enhance the convergence rate and stability of the algorithm, becoming one of the effective global optimization methods.

Though HGA algorithm has very good results for easy and challenging Sudoku puzzles, it is not so ideal for difficult and super difficult Sudoku puzzles. This awaits our deeper study on this algorithm in the future work to make it more suitable to solve the difficult and super difficult Sudoku puzzles.

Acknowledgments The authors wish to acknowledge the anonymous reviewer for valuable comments and suggestions, which helped to improve the article. This research was supported by the National Natural Science Foundation of China (No. 60974077), the Science Technology Project of Guangdong Proving (No. 2011B010200042) and the Natural Science Foundation of Guangdong Proving (No. 10251009001000002).

References

1. Duan, X.D., Wang, C.R., Liu, X.D.: Particle Swarm Optimization and Application, pp. 145–152. Liao Ning University Press, Shen Yang (2007)
2. Geem, Z.W.: Harmony search algorithm for solving sudoku. Lect. Notes Comput. Sci. **4692**, 371–378 (2007)
3. Holland, J.H.: Adaptation in Natural and Artificial Systems, 2nd edn. University of Michigan Press, Ann Arbor (1992)
4. http://en.wikipedia.org/wiki/Sudoku. Accessed 16 Oct 2006
5. http://www.llang.net/sudoku/

6. Kennedy, J., Eberhart, R.: Particle Swarm Optimization. In: Proceedings of IEEE International Conference on Neural Networks. Perth, pp. 1942–1948 (1995)
7. Li, H.: Algorithm and implementation for Sudoku puzzle based on graph search algorithm. J. Tonghua Nomal Technol. **30**(10), 43–45 (2009)
8. Li, Y.D., Deng, X.Q.: Solving Sudoku puzzles base on improved genetic algorithm. Comput. Appl. Softw. **28**(3), 68–70 (2011)
9. Moraglio, A., Togelius, J., Lucas, S.: Product geometric crossover for the Sudoku puzzle. In: 2006 IEEE Congress on Evolutionary Computation (CEC2006), Vancouver, BC, Canada, July 16–21, pp. 470–476 (2006)
10. Moraglio, A., Togelius, J.: Geometric Particle Swarm Optimization for the Sudoku Puzzle. In: Genetic and Evolutionary Computation Conference London, pp. 118–125 (2007)
11. Mantere, T., Koljonen, J.: Solving, Rating and Generating Sudoku Puzzles with GA. 2007 IEEE Congress on Evolutionary Computation-CEC2007, Singapore, pp. 1382–1389 (2007)
12. Mantere, T., Koljonen, J.: Solving and analyzing Sudokus with cultural algorithms. 2008 IEEE Congress Computational Intelligence—WCCI2008, 1–6 June, Hong Kong, China, pp. 4054–4061 (2008)
13. Mantere, T., Koljonen, J.: Ant Colony Optimization and a Hybrid Genetic Algorithm for Sudoku Solving. In: 15th International Conference on Soft Computing, Brno, Czech Republic, Mendell 2009, pp. 41–48 (2009)
14. Nicolau, M., Ryan, C.: Genetic operators and sequencing in the GAuGE system. In: IEEE Congress on Evolutionary Computation CEC 2006, 16–21 July, pp. 1561–1568 (2006)
15. Nicolau, M., Ryan, C. : 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. Springer, Heidelberg (2006)
16. Perez, M., Marwala, T.: Stochastic optimization approaches for solving Sudoku. In: Computer Science—Neural Evolutionary Computing (2008). arXiv: 0805.0697v1. http://arxiv.org/ftp/arxiv/papers/0805/0805.0697.pdf
17. Semeniuk, I.: Stuck on you. NewScientist **24**(31), 45–47 (2005)
18. Sullivan, F.: Born to compute. Comput. Sci. Eng. **8**(4), 88–90 (2006)