Solving and Analyzing Sudokus with Cultural Algorithms

Timo Mantere & Janne Koljonen

University of Vaasa
Department of Electrical Engineering and Automation
P.O. Box 700,
FIN-65101 Vaasa, Finland

timan@uwasa.fi & jako@uwasa.fi
www.uwasa.fi/~timan/sudoku

Outline

- Introduction to Sudoku problem
- Encoding of the Sudoku problem in our EAs
  - Crossover and mutations operators
  - Fitness function
  - Belief space
- Results and their interpretation
  - Comparison of results
- Conclusions
- Future

Introduction

- This paper studies the problems involved in solving and analyzing Sudokus with cultural algorithms
- Sudoku is a Japanese logical game that has recently become hugely popular in Europe and North-America. However, the first puzzle was published in a puzzle magazine in USA 1979, then it circled through Japan, where it became popular in 1986, and later it become a phenomenon in the western world circa 2005.
- Sudoku has been claimed to be very popular and addictive because it is very challenging but has very simple rules.

- The objectives of this study were
  - 1) to test if a cultural algorithm (**CA**) with a belief space solves Sudoku puzzles more efficiently than a normal permutation genetic algorithm (**GA**),
  - 2) to see if the belief space gathers information that helps analyze the results and improve the method accordingly,
  - 3) to improve our previous Sudoku solver presented in CEC2007.

Solving and Analyzing Sudokus with Cultural Algorithms 5/30/2008

A Sudoku puzzle example

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

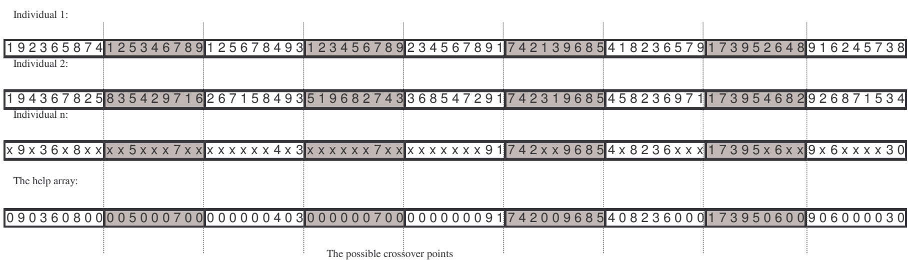
- Sudoku puzzle is composed of a 9×9 grid, that is divided into nine 3×3 sub grids.
- The solution of Sudoku puzzle is such that each row, column and sub grid contains each integer from [1, 9] once and only once.
- In addition, there are some static numbers ("givens") that must stay in their fixed position

A Sudoku puzzle example

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

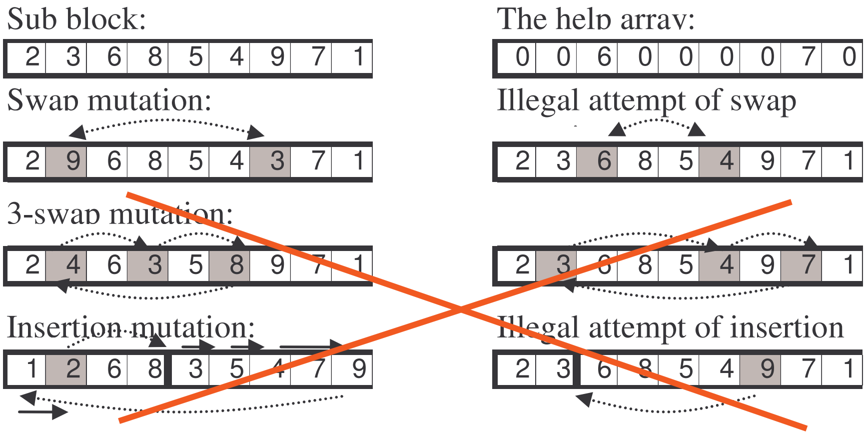
- The Sudoku solution must be unique (usually)
- Note that each column, row and sub square of the solution contains each integer from 1 to 9 once
- The **givens** given in the beginning are in their original positions. Other positions have been solved.
  - The number of givens does not determine the difficulty. Grading puzzles is one of the most difficult things in Sudoku creation, and there are approx. 15-20 factors that have an effect on the difficulty rating

Genetic encoding



- The representation of Sudoku puzzles with our GA & CA
- One individual is an array of 81 numbers, which is divided into nine sub blocks of nine numbers
  - The allowed crossover points are only between sub blocks (marked as vertical lines)
  - The help array is used for checking fixed positions: if there is a number that is not equal to zero, that number cannot be changed

Swap mutation

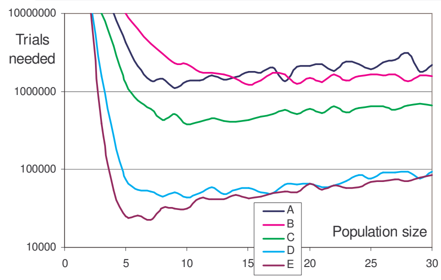


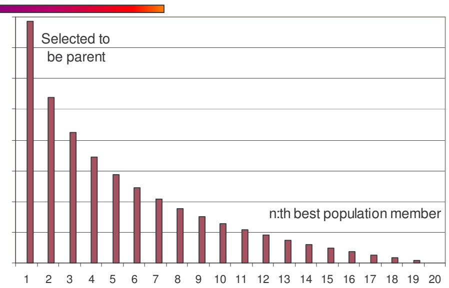
- The mutation types used in the Sudoku optimization (~~removed from this version~~)
  - Up left; one sub block, up right; the givens in that sub block (6 and)
  - The mutation is applied so, that we randomly select positions inside the sub block, and then check the help array if the positions are free to change

GA parameters

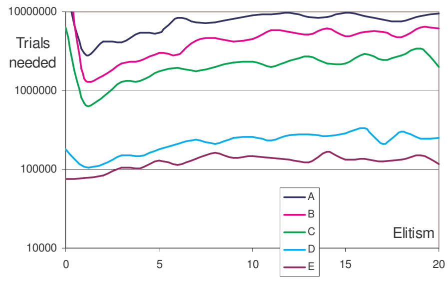
- The selected parameters for the test runs were the following:
  - Combinatorial GA,
    - Chromosome consist of 81 integer numbers
    - Uniform crossover with fixed crossover points (9-point crossover)
  - Population size N=11,
  - Elitism N_e=1
  - Mutation percentage was measured to be 3.7% per one Sudoku puzzle location
  - Swap mutation sequences with 1-5 swaps with percentages {62.5, 30.4, 6.6, 0.5, 0.01}
  - Crossover ratio 100
    - New individuals generated by first doing crossover and then mutation to the new trial. We measured that 88.5% of new individuals have been changed by mutation and 11.5% only by crossover
  - Stopping condition was solution found
    - The most difficult Sudoku with the worst test run required 10 394 690 trial evaluations

GA parameters





Reasoning for population size (up) and elitism (below)



We favored the best individuals as parents by selecting the mating individuals p1 and p2 with using the following Java code:

```
for(i=POP-1; i>=ELIT; i--){
        ii=ord[i];
        p1 = ord[i*Math.random()];
        p2 = ord[i*Math.random()];
        crossover(indiv[ii], indiv[p1], indiv[p2]);
        mutation(indiv[ii]);
        }
```


Fitness function

The fitness function was composed of three parts

- The first part requires that all digits {1,…,9} must be present in each row and column, otherwise penalty P_x is added

$$P_x = \sum_{i=1}^{8}\sum_{j=1}^{8}\sum_{ii=i+1}^{9}\sum_{jj=j+1}^{9}\left[\left(x_{i,j} == x_{ii,j}\right) + \left(x_{i,j} == x_{i,jj}\right)\right]$$

- The second part is aging of the best individual (adding 1 to its fitness value each round when it remains the best)

```
if (Best[generation[i]] == Best[generation[i-1]])
        Value[Best]+=1;
```

- The third part requires that the same digit as some **given** must not appear in the same row or column as a **given**, otherwise penalty P_g added
  - This used only after reaching the near solution region of the search space

$$P_g = \sum_{i=1}^{9}\sum_{j=1}^{9}\left(x_{ij} == g_{ij}\right)$$

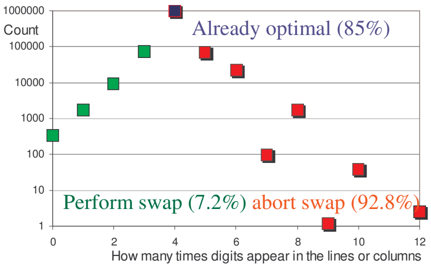
Analysis of operators

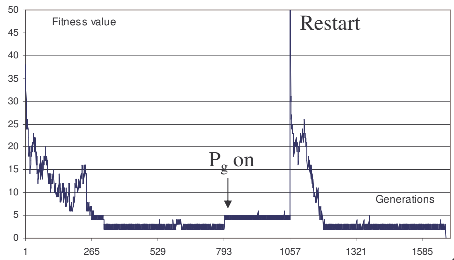
8	4	6	9	1	3	7	5	2
3	9	5	7	4	2	8	1	6
1	7	2	6	5	8	9	3	4
7	5	4	2	8	9	1	6	3
9	8	3	1	6	5	2	4	7
6	2	1	4	3	7	5	8	9
2	3	8	5	9	4	6	7	1
4	1	9	8	7	6	3	2	5
5	6	7	3	2	1	4	9	8

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

Sudoku	*Our version*	*Without aging*	*Tighter <3 limit*	*Looser <5 limit*
E*b*	271	290	1315	1280570
E*c*	549	702	2173	3209492
E*a*	553	850	3739	4449784
1*b*	665	1070	6318	5703074
1*a*	1141	3461	24170	*Too many*
GA-E*c*	3057	7700	13582	11033764
3*a*	17123	179608	5826266	*Too many*
2*b*	33116	781994	1694321	*Too many*
5*a*	62813	950079	*Too many*	*Too many*
SD*a*	422542	5969736	*Too many*	*Too many*

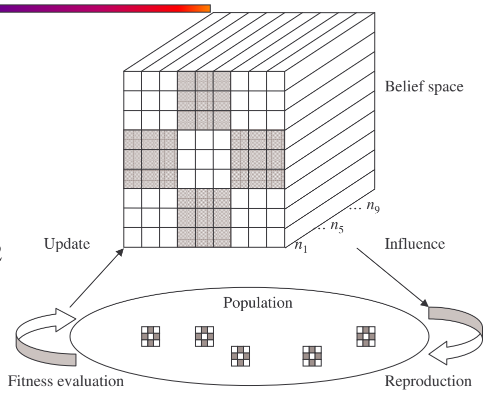
Aborting swap attempts, if it would lead too many identical digits in the rows or columns





Belief space

- The belief space in this case was a 9×9×9 cube, where the first two dimensions correspond to the positions of a Sudoku puzzle, and the third dimension represents the nine possible digits for each location
- After each generation, the belief space is updated if:
  1) The fitness value of best individual is 2
  2) The best individual is not identical with the individual that updated the belief space previous time
- The belief space is updated so that the value of the digit that appears in the best Sudoku solution is incremented by 1 in the belief space.
  - This model also means that the belief space is updated only with near-optimal solutions (2 positions wrong)
  - This information is used only in the population reinitialization process



When population is reinitialized, positions that have only one non-zero digit value in the belief space are considered as givens, these include the real givens and also so called “hidden” givens that the belief space have learned, *i.e.* those positions that always contain the same digit in the near-optimal solutions

Givens and ‘hidden givens’

How many givens; real *G* and hidden *H*, each of our benchmark Sudoku instance had + the amount of different near solution *N* we found

Those marked with * actually posses zero hidden givens, when analyzed of all 100 solve runs

Diff rating.	*Givens (G)*			*Hidden givens (H)*			*Near solutions (N)*		
	a	*b*	*c*	*a*	*b*	*c*	*a*	*b*	*c*
1	33	36	32	34	34	15	41	16	51
2	30	28	28	27	11	16	46	58	64
3	28	26	27	17	14	7	89	116	107
4	28	27	28	9	11	* 7	88	119	123
5	30	28	26	11	3	8	126	118	234
E	36	39	36	32	35	33	21	8	8
C	25	25	25	19	10	11	99	126	122
D	22	23	22	* 5	11	18	319	118	89
SD	23	22	22	6	* 13	11	140	249	147
Easy	31	31	32	20	32	21	36	50	50
Med	28	26	28	11	8	19	118	140	174
Hard	22	26	23	* 2	6	* 5	263	107	198
GA-E	32	33	37	12	17	19	62	35	38
GA-M	29	32	31	8	11	10	63	104	96
GA-H	27	27	24	10	10	7	174	136	148

Example of near solutions

Correct:

```
 2  9  4 |  8  6  3 |  5  1  7 =  0
 7  1  5 |  4  2  9 |  6  3  8 =  0
 8  6  3 |  7  5  1 |  4  9  2 =  0
--------------------
 1  5  2 |  9  4  7 |  8  6  3 =  0
 4  7  9 |  3  8  6 |  2  5  1 =  0
 6  3  8 |  5  1  2 |  9  7  4 =  0
--------------------
 9  8  6 |  1  3  4 |  7  2  5 =  0
 5  2  1 |  6  7  8 |  3  4  9 =  0
 3  4  7 |  2  9  5 |  1  8  6 =  0

 0  0  0  0  0  0  0  0  0
45 45 45 45 45 45 45 45 45
```

Near sol. (2 wrong) Pos. wrong: 42

```
 9  8  4 |  2  6  3 |  5  1  7 =  0
 2  1  5 |  7  4  9 |  6  3  8 =  0
 7  6  3 |  1  8  5 |  4  9  2 =  0
--------------------
 1  3  8 |  4  2  7 |  9  6  5 =  0
 4  5  9 |  8  1  6 |  2  7  3 =  0
 6  7  2 |  5  3  9 |  1  8  4 =  0
--------------------
 8  9  1 |  3  5  4 |  7  2  6 =  0
 5  2  7 |  6  9  8 |  3  4  1 =  0
 3  4  6 |  2  7  1 |  8  5  9 =  0

 0  0  0  1  0  1  0  0  0
45 45 45 38 45 52 45 45 45
```

Near sol. (2 wrong) Pos. wrong: 20

```
 4  9  2 |  8  6  3 |  5  1  7 =  0
 7  1  5 |  2  4  9 |  6  3  8 =  0
 8  6  3 |  7  5  1 |  4  9  2 =  0
--------------------
 1  5  4 |  9  2  7 |  8  6  3 =  0
 6  7  9 |  4  3  8 |  2  5  1 =  0
 2  3  8 |  5  1  6 |  9  7  4 =  0
--------------------
 9  8  6 |  3  7  4 |  1  2  5 =  0
 5  2  7 |  6  8  1 |  3  4  9 =  0
 3  4  1 |  2  9  5 |  7  8  6 =  0

 0  0  0  1  0  1  0  0  0
45 45 45 46 45 44 45 45 45
```

Near sol. (2 wrong) Pos. wrong: 39

```
 4  9  3 |  8  5  3 |  6  1  7 =  1
 2  1  5 |  7  6  9 |  4  3  8 =  0
 8  6  7 |  1  4  2 |  9  5  2 =  1
--------------------
 1  5  4 |  9  2  7 |  8  6  3 =  0
 6  8  9 |  3  1  4 |  2  7  5 =  0
 7  3  2 |  5  8  6 |  1  9  4 =  0
--------------------
 9  7  1 |  4  3  8 |  5  2  6 =  0
 5  2  8 |  6  7  1 |  3  4  9 =  0
 3  4  6 |  2  9  5 |  7  8  1 =  0

 0  0  0  0  0  0  0  0  0
45 45 45 45 45 45 45 45 45
```

Near sol. (2 wrong) Pos. wrong: 38

```
 2  9  8 |  4  5  3 |  6  1  7 =  0
 7  1  5 |  1  6  9 |  4  3  8 =  1
 4  6  3 |  7  8  2 |  9  5  2 =  1
--------------------
 1  3  4 |  9  2  7 |  5  8  6 =  0
 6  5  9 |  8  1  4 |  2  7  3 =  0
 8  7  2 |  5  3  6 |  1  9  4 =  0
--------------------
 9  8  6 |  3  4  1 |  7  2  5 =  0
 5  2  1 |  6  7  8 |  3  4  9 =  0
 3  4  7 |  2  9  5 |  8  6  1 =  0

 0  0  0  0  0  0  0  0  0
45 45 45 45 45 45 45 45 45
```

Near sol. (2 wrong) Pos. wrong: 32

```
 2  8  4 |  6  5  3 |  9  1  7 =  0
 7  1  5 |  4  2  9 |  6  3  8 =  0
 9  6  3 |  1  7  8 |  4  5  2 =  0
--------------------
 1  5  2 |  9  4  7 |  8  6  3 =  0
 4  3  9 |  8  1  6 |  2  7  5 =  0
 8  7  6 |  5  3  2 |  1  9  4 =  0
--------------------
 6  9  8 |  3  7  4 |  5  2  1 =  0
 5  2  7 |  6  8  1 |  3  4  9 =  0
 3  4  1 |  2  9  5 |  7  8  6 =  0

 0  0  0  1  1  0  0  0  0
45 45 45 44 46 45 45 45 45
```

Results (1st version of CA)

Difficulty Rating	Average of Solve generations with GA *a*	*b*	*c*	Average of Solve generations with CA *a*	*b*	*c*	Improve by %
1	78	41	1244	**76**	**41**	**1191**	-4.08
2	1203	6951	2328	**831**	**6230**	**2016**	-13.40
3	**2103**	9517	5835	2207	**8238**	**5062**	-11.16
4	**5503**	10966	**9369**	6483	**9618**	10115	1.46
5	8371	8661	**13649**	**7155**	**7698**	15808	-0.06
E	40	23	56	**36**	**20**	**48**	-12.65
C	5144	**10418**	7010	**4866**	11185	**5704**	-3.62
D	40830	**19486**	8433	**40162**	19850	**8255**	-0.70
SD	**39901**	20593	27918	42841	**20095**	**27416**	2.19
Easy	1669	797	**500**	**1386**	**791**	503	-9.69
Med	14576	21740	5660	**14317**	**19468**	**5431**	-6.58
Hard	**125105**	**11629**	48479	125391	11875	**45529**	-1.31
GA-E	771	**339**	798	**684**	423	**584**	-11.40
GA-M	4501	4253	3947	**4418**	**3483**	**3390**	-11.10
GA-H	**16528**	11354	62588	20034	**11055**	**53074**	-6.97

- The first version of CA (in the submitted draft paper) was slightly different than the one represented in these slides
- This CA version was more aggressive and obtain better solutions with easier Sudoku instances
- This CA was 2.6% more efficient than GA
  - With difficult Sudokus it performed poorly, which weighted down the advantage

Results (this version)

Difficulty Rating	Average amount of trials needed to solve with GA			Average amount of trials needed to Solve with CA			Improve by
	a	*b*	*c*	*a*	*b*	*c*	%
1	1264	**634**	4787	**1141**	665	**4732**	2.19
2	8765	**32618**	**12828**	**8187**	33116	14515	-2.97
3	17841	70214	**35450**	**17123**	**65068**	42332	-0.82
4	**47057**	70994	71539	50083	**67691**	**68229**	1.89
5	66813	**49802**	**101691**	**62813**	54625	114180	-6.10
E	**535**	**252**	600	553	271	**549**	1.05
C	**24656**	**80486**	50406	26330	84761	**41034**	2.20
D	281519	90496	**66810**	**250518**	**83608**	71503	7.56
SD	**413450**	241184	218102	422542	**222883**	**207893**	-0.22
Easy	11261	2976	**3340**	**11109**	**2800**	3520	0.84
Med	66183	**191627**	**53365**	**63676**	199871	53806	-1.99
Hard	1419023	90883	627091	**1232282**	**81677**	**530257**	13.70
GA-E	4128	**4523**	3100	**4065**	4596	**3057**	9.31
GA-M	36735	19186	32651	**33808**	**17242**	**29536**	9.02
GA-H	**163636**	**104389**	785814	193622	104655	**601404**	14.63
Sum		5680705			**5187928**		8.67

Interpretation of results

We calculate the correlations between Sudoku difficulty and some numbers calculated from the Sudoku or with the help of belief space

Highest correlations:

1) CA results and the overall number of the near-optimal solutions that a Sudoku instance possesses N_{all}
   - The number of near-optimal N_{all} solutions is the most important factor to define Sudoku puzzle difficulty, However, N_{all} is unknown during a Sudoku solve run. Thus it cannot be employed in the optimization (it is counted from the series of 100 solve runs)
2) Number of givens G and CA solving efficiency
   - This means that although the number of givens does not at implicitly define the difficulty of the Sudoku, it has large influence

- The amount of hidden givens does not have high correlation with the results
- The number of hidden givens adjusted with the number of free locations $Ha=H/(81-G)$ in the Sudoku does not explain results better than unadjusted

	GA	*CA*	*Improv.*
CA	0.996		0.391
G	-0.512	-0.533	-0.168
N	0.462	0.486	0.111
N_{all}	0.601	0.624	0.187
M_{avg}	0.501	0.516	0.109
H	-0.439	-0.457	-0.099
H_{all}	-0.382	-0.410	-0.115
H_{avg}	-0.205	-0.224	0.014
Ha_{min}	-0.420	-0.438	-0.103
Ha_{all}	-0.371	-0.397	-0.117
Ha_{avg}	-0.410	-0.435	-0.085

Some Sudokus were found to posses zero hidden givens. These are quite difficult since all free positions can have different values in some of the near-optimal solutions

Comparison of the results

Sudoku problems from www.sudoku.com	*Our GA*		*The best results represented by Moraglio ea*		
	Unlimited trials	100000 trials	Hamming space crossovers	Swap Space crossovers	Hill climbers
Easy 1	30	30	5	28	30
Easy 2	30	30	8	21	30
Easy 3	30	30	14	30	30
Medium	30	22	0	0	0
Hard	30	2	0	15	0
Total	150	114	27	94	90

Our results and the best results represented by Moraglio *et al* in each of the three difficulty categories of Sudoku's found from www.sudoku.com. The numbers represents how many times out of 30 test runs each method reached the optimum with each problem.

Conclusions

- The results show that EAs are fairly effective to solve Sudoku puzzles (however, not the fastest methods)
- CA is just slightly more efficient than GA, and CA seems to work better with the most difficult puzzles
- Our results stand quite well the comparison with the other known results with EAs (see the paper)
- The lack of common benchmark Sudokus complicates the comparison of results
  - We decided to put our 46 test Sudokus available in the web, so that anyone interested to compare their results with ours can now use the same benchmark puzzles

    http://www.uwasa.fi/~timan/sudoku

Conclusions

- The difficulty ratings given for Sudoku puzzles in newspapers seems to be consistent with their difficulty in GA optimization. For some solitary puzzles the rating seems to be wrong, but the overall trend follows the ratings
  - This means that GA can be used to rate the difficulty of a new Sudoku puzzle
  - However, the other explanation can be that the original puzzles are also generated with computer programs, and since GA is also a computer based method, it is possible that a human solver does not necessarily experience difficulty the same way

Conclusions

- When some belief spaces were analyzed manually, it looked like Sudoku puzzles might possess some kind of positional bias
  - Most of the belief spaces looked like the trials composed based on them would more likely contain small numbers in the left upper corner and larger numbers in right bottom corner
  - We think that it is possible that Sudoku generators have some kind of positional bias when they generate new Sudoku puzzles
  - CA belief space could potentially exploit this bias in order to generate better results.
  - We plan to measure the possible positional biases in future and see, if it really appears or not, and if it appears only with some Sudoku generators

Future, Ants?

- The belief space model used in this study was quite simple and can possibly be improved in future. It is likely that the gathered information could be employed more efficiently than just in the reinitialization
- The CA might also be improved by some kind of energy function based belief space
- Lately, we have solved Sudokus with Ant colony optimization (ACO)
  - Results showed that ACO is more effective than CA with 26/46 benchmark Sudokus (including 21 easiest), but it fails to find solution efficiently with 11 difficult instances
  - We are considering some kind of ACO/GA hybrid (cultural part is embedded to the ACO pheromone matrix)